AN ACCESS CONTROL MODEL FOR THE UNIFRAME FRAMEWORK

A Thesis
Submitted to the Faculty
of
Purdue University
by
Alexander M. Crespi

In Partial Fulfillment of the
Requirements for the Degree
of
Master of Science

May 2005

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **MAY 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **An Access Control Model for the Uniframe Framework**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Indiana University/Purdue University,Department of Computer and Information Sciences,Indianapolis,IN,46202**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)
12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**		
13. SUPPLEMENTARY NOTES		
14. ABSTRACT **see report**		
15. SUBJECT TERMS		

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**304**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

To Eunbok.

ACKNOWLEDGEMENTS

The graduate education at the Department of Computer and Information Science, Indiana University-Purdue University, has given me a strong foundation in computer science as I begin my new career in software development. I would like to thank the many people who have supported me during my work on this degree program and thesis.

First, I would like to thank my advisor Dr. Raje for is support and guidance throughout my graduate studies. He challenged me to stretch my thinking as I did research for this thesis and his deep knowledge has enriched my understanding of computer science and helped me develop more perseverance and critical judgment.

I extend my thanks to Dr. Andrew Olson and Dr. Xukai Zou for being on my advisory committee and for their efforts in critiquing my thesis research.

I would also like to thank Carol Burt from 2AB, Inc. for the time she took to answer my questions concerning model driven access control. I very much enjoyed our correspondence and appreciated her help.

I thank all of my fellow graduate students working on the UniFrame project. I have enjoyed working with each of them over the course of my studies and will miss their insight and antics. I wish them each the best in their future endeavors.

I want to thank the U.S. Department of Defense and the U.S. Office of Naval Research for supporting the UniFrame research project under award number N00014-01-1-0746.

I give thanks for my parents, Henry and Mary Crespi, for instilling in me a love of learning that pushes me to this day to ask questions and explore the unknown.

Finally, I give my deepest thanks to my wife Eunbok for her love, encouragement, and support. She is my eternal sunshine.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

Table Page

LIST OF FIGURES

ABSTRACT

Crespi, Alexander M., M.S., Purdue University, May 2005. An Access Control Model for the UniFrame Framework. Major Professor: Dr. Rajeev Raje.

Component-based software development, while offering a potential solution for the creation of complex distributed systems, requires a framework for specifying component properties such that the behaviors of a system may be analyzed before composition and verified during operation. With much energy expended on verifying security properties of software systems, a means of composing a system's security characteristics from the properties of individual components would aid in the creation of more secure systems. In this thesis, a framework for characterizing the access control properties of distributed software components along with a compositional model for predicting system characteristics are presented. The proposed framework will address the following issues: a) development of a means of specifying access control properties for individual components and integrated systems, b) extension of the UniFrame Resource Discovery Service to facilitate searching for components with required access control properties, and c) the creation of composition models for predicting the system behavior with respect to access control properties and access control policies. Component and system specifications are based on logic programming and Temporal Logic of Action in order to provide a means for model-checking and verification. A simple student information system case study is used as the context for describing and testing this access control framework.

1. INTRODUCTION

As evermore computers are interconnected via high speed networks, the computing world is shifting from centralized to distributed computing systems (DCS). This shift coincides with the growth of object-oriented programming (OOP) which has made code reuse more practicable. As an extension of OOP, component-based software development (CBSD) has taken the concept of reuse a step further by advocating the development of self-contained software entities that have a public interface and private implementation. These software components exist independently but can be composed into coalitions of components cooperating to achieve higher level tasks. The combination of DCS and CBSD has resulted in a proliferation of distributed computing models (CORBA, J2EE, .NET, etc.) that necessitates interactions between heterogeneous software components.

The UniFrame Approach [RAJ01, RAJ02] provides a framework for the seamless interaction of heterogeneous software components. There are three overall aspects that are united in Uniframe: architecture-based interoperability, distributed resource discovery, and validation of quality requirements. Architecture-based interoperability addressed issues of automation and standardized approaches when it comes to combining heterogeneous component technologies into a single integrated system. Distributed resource discovery focuses on system and component specifications, distribution of the component search mechanism, and selection of components that meet system requirements. Finally, validation of quality requirements examines the vocabulary, metrics, and methods used to determine if components and integrated systems indeed meet specified quality requirements. Through the UniFrame Approach and its associated tools, distributed computing systems can be constructed in a semi-automated fashion facilitating the creation of DCS with predictable QoS characteristics.

1.1 Problem Definition and Motivation

Quality of Service encompasses a wide variety of concepts ranging from throughput and end-to-end delay to availability and security. The QoS characteristic of security is quite complex in itself and the importance of security in computing systems has increased dramatically with the advent of networks and distributed computing. With evermore private and sensitive information being stored and shared via networks, the means of controlling accesses to computing resources has grown in importance. This is especially true for current attempts to make medical records digital. The Common Criteria [CC04] provides a framework for creating functional security requirements and for evaluating the level of assurance that a system meets the required security requirements. One deficiency of the Common Criteria is its inability to support composition of specifications for systems created from multiple certified components. Currently, such a composition would result in the creation of a new set of specifications for the composed system and then a re-evaluation of the assurance level for the system. Ideally, it would be productive to have an automated method for assessing the security assurance of a composed system from the certification information from the individual components. However, an automated method for composing specifications in order to facilitate the evaluation of the security assurance level of the composed system would aid in the process on evaluating the composed system.

To facilitate the automated composition of distributed component systems using the UniFrame Approach, it is necessary to develop a mechanism for describing the security characteristics of the desired system. Such system characteristics must then be decomposed into the desired component properties to facilitate searching for software components that can be used to compose the integrated system. It should then be possible to statically test the proposed integrated system to determine if selected components do indeed create a system that meets the original security requirements or not.

One important subset of security properties is access control. In many modern computing systems, complex access control policies are often coded directly into the application code. In environments where policy requirements may change due to regulations or laws, this may mean that software applications can require frequent and sometimes major modifications to these policies. The Model Driven Access Control [BUR03] proposes a model for separating the access control architecture from the application code, as much as possible, to facilitate the separation of concerns. In this model, changes in policy requirements are less likely to require major changes to the application code. This thesis explores only the access control portion of security features in the context of UniFrame.

1.2 Objectives: Statement of Goals

The specific objectives of this thesis are:

- To provide a means to express the access control characteristics of distributed computing systems in the UniFrame paradigm.
- To provide a means of expressing the access control characteristics of individual software components.
- To create a means of identifying individual software components that meet system access control requirements.
- To provide a means of statically predicting the access control behavior of an integrated system based on the properties of individual components.

1.3 Contributions of this Thesis

In order to facilitate the automated assembly of distributed component-base computing systems, this thesis will define a means for characterizing, decomposing, and composing the access control security characteristics of distributed software components.

- Provides an access control framework for the UniFrame Approach.
  - Extended the UMM Specification for access control properties.
  - Created a matching algorithm for extending Resource Discovery to include access control properties.
  - Extended Knowledge-Base to include access control related information.
    - Resource definition and naming.

    - TLA system model framework.
    - Prolog system model framework.
    - Predicate naming conventions.
- Static verification of access control policy implementation using TLA.
- Prediction of effects of access control policy on system dynamics.
- Prediction of effects of component access control properties on system behavior.
- Evaluate changes in access control policy before implementation.
- Prediction of user access privileges to system level functionality.

1.4 Organization of this Thesis

This thesis is organized into eight chapters. Chapter 1 provides an introduction to the thesis including defining the problem and motivations for the research along with goals and contributions made. Chapter 2 presents previous and related works. Chapter 3 provides an introduction to the student information system case study developed in this thesis. Chapter 4 discusses characterizing the access control properties of a distributed computing system as well as individual software components in the context of the UniFrame Approach. Chapter 4 also addresses augmentation of the UniFrame Resource Discovery Service to incorporate searching for access control related characteristics. Chapter 5 investigates using logic programming to predict access control characteristics of distributed computing systems. Chapter 6 describes how to use TLA to verify the functionality of the system relative to access control influenced characteristics. Chapter 7 examines the case study performed to provide a real-world test for the ideas presented in this thesis. Finally, Chapter 8 concludes the thesis with a summary of the work and possible future extensions.

2. BACKGROUND AND RELATED WORK

This chapter provides background on the various concepts and ideas used throughout this thesis. Section 2.1 provides an overview of the UniFrame Approach, which this thesis extends to incorporate access control security QoS properties. Section 2.2 introduces the concepts behind access control and various existing access control models. Section 2.3 generalizes the discussion of access control by introducing model driven access control. Section 2.4 examines the Common Criteria security assurance framework and describes how it relates to access control. Section 2.5 provides an overview of several access control models for distributed systems. Finally, Section 2.6 compares the utility of Petri nets and the Temporal Logic of Action in modeling access control in distributed systems.

2.1 UniFrame

As stated earlier, the UniFrame Approach provides a framework for a seemless integration of heterogeneous distributed software components [RAJ00, RAJ01, RAJ02]. To the extent possible, UniFrame has the goal of facilitating automation of the integration of the components while being aware of the quality of service of the individual components as well as a system composed from these components. One aspect of the quality of service for components and composed systems is the ability to regulate access to resources under its control. In order to address access control as a quality of service issue, it is necessary to understand the structure of UniFrame. The following sections provide an overview of the UniFrame Approach including the Unified Meta-component Model (UMM), the Quality of Service Framework, and the UniFrame Resource Discovery Service (URDS).

2.1.1 Unified Meta-component Model (UMM)

The Object Management Group (OMG) has recently shifted its focus to Model Driven Architecture (MDA) in an effort to provide a standardization of Business and Component Meta-Models in order to facilitate the creation of mechanized software for the collaboration and bridging of component architectures [OMG01a]. The unified meta-component model supplies a means for bridging the gaps between different component models (CORBE, J2EE, COM, etc.) by providing a formal meta-model that can be used to interrelate component models. The UMM specifications of components consist of three core parts: components, service and service guarantees, and infrastructure.

2.1.1.1 Components

UMM components are autonomous entities with non-uniform implementations with each component adhering to some distributed-component model and no unified implementation framework. Components are expected to have well-defined interfaces and private implementations. In UMM, components have three aspects: computational aspect, cooperative aspect, and auxiliary aspect. Computational aspects describe the tasks carried out by each component and in order for components to understand the functionality of other components, UMM supports reflection. Cooperative aspects describe how components cooperate with each other and depend on factors such as cost of service, inter-component negotiations, aggregation, quality, and duration. Auxiliary aspects may include properties such as mobility, security, and fault tolerance.

2.1.1.2 Services and Service Guarantees

In distributed computing systems it is natural to have multiple choices for obtaining a specific service. Therefore each component, in addition to specifying functionality, must also specify the level of quality of service that can be expected. The quality of service of a component provides the system integrator with some level of assurance that a component will operate satisfactorily during its deployment in a distributed computing system. This could include such information as the algorithm used to carry out a task, resource required, and expected computational effort. The UniFrame

Quality of Service Framework (UQoS) [BRA01] addresses the guarantees of QoS at both the component and system levels.

2.1.1.3 Infrastructure

The UMM provides for a discovery infrastructure for locating components based on functional and QoS characteristics. This architecture is primarily made up of two types of entities: headhunters and Internet Component Brokers. Headhunters are responsible for detecting the presence of new components in its search space, storing component functionality, and matching component characteristics to search queries. The Internet Component Broker (ICB) mediates between two components adhering to different component models through adapter technology. Interoperability is achieved through wrap and glue technology.

2.1.2 UniFrame Approach

The UniFrame Approach is best explained through the process outlined in the Figure 2-1. It is addressed in two aspects: a) component development and deployment, and b) system integration. The UniFrame Knowledgebase, at the center of the figure and of the key importance to both aspects of the process, represents the overall domain model for the systems being modeled with UniFrame and is created by domain experts while consulting industry standards for that domain. The component developer will craft components based on the knowledgebase standards, the quality of the component will be described by various measures, and then the component is deployed as an active component ready for integration into a distributed computing system.

Assuming that there are many deployed components, the system integrator will submit a query for a system with certain functional and QoS constraints. This query will be decomposed into queries for the individual component types needed to assemble the system. This modified query will be submitted to the UniFrame Resource Discovery Service (URDS) which in turn searches, selects, and returns a list of components that will meet the functional and nonfunctional requirements needed for components in the system being created [SIR02]. The system integrator then selects components and assembles the system which is then tested for quality. If the system passes the quality checks, it is then deployed. Otherwise the system integrator selects other components or modified the initial query to account for new properties.

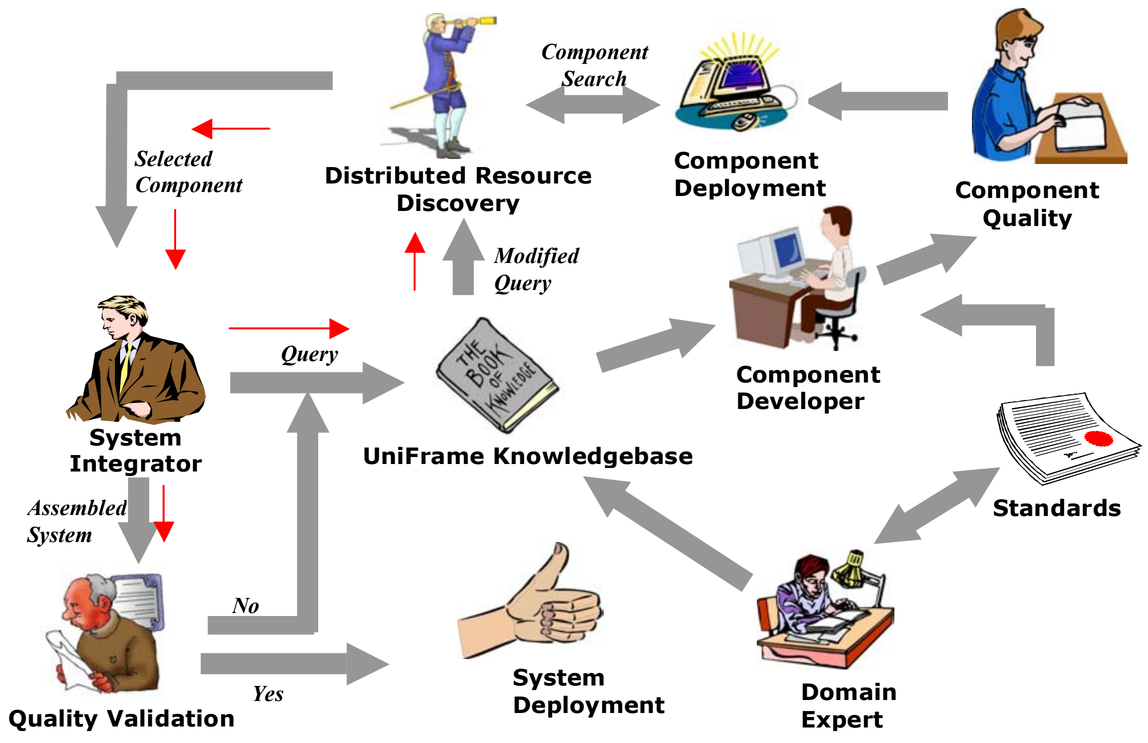

Figure 2-1: UniFrame Approach

A key component of the UniFrame approach, the URDS itself is a distributed system consisting of the Query Manager, Headhunters, and Active Registries. An active registry advertises deployed components from a single component model (JAVA, CORBA, etc.) and the search space can contain many different active registries each cataloging many deployed components. Headhunters are autonomous programs that search active registries and create their own individual repositories of information concerning deployed components. A headhunter may specialize in the types and qualities of the components it is willing to represent. The Query Manager takes in a query from the system integrator and send it onto the headhunters which attempt to match the query requirements to their own component records. If a match is found, the result is returned to the Query Manager. The Query Manager collects all of the matching results from headhunters and then returns these matches to the system integrator [SIR02].

In this thesis, several aspects of UniFrame are augmented to handle access control security QoS issues. First, the UMM auxiliary aspect, which includes security, is augmented to include access control specifications for components. Second, the knowledge-base is expanded to include composition and decomposition rules for access control properties. Third, the URDS is extended to incorporate access control properties into matching algorithms. Finally, quality validation expands to include testing of access control properties of the composed system.

2.2 Access Control

In order to consider what the access control properties of a component might look like, it is important to examine the variety of existing access control models. At the most general level, an access control enforcement is carried out by a reference monitor, also referred to as a guard, which consults an authorization database to determine if a subject is authorized to perform an operation on an object [SAN94]. The authorization database may be may be stored with the objects being protected, as in the case of access control lists, stored with the subjects in the system, as with capabilities; or even stored in some centralized repository. The following sections explore different schemes used to implement access control mechanisms and discuss some of the advantages and disadvantages of the models.

2.2.1 Access Matrix

The basic model for access control is the access control matrix, which consists of a two dimensional matrix relating subjects to objects. Each cell in the matrix contains the access privileges of one subject for accessing one object [SAU01]. For large systems, the access control matrix can be extremely large and consist of mostly empty cells leading to great inefficiencies in implementation [SAN94]. Therefore, most systems implement access control models that can be mapped back to the concept of the access control matrix but avoid these inefficiencies.

2.2.2 Access Control Lists

Access Control Lists (ACL) are one means of implementing access control matrices while avoiding the inefficiencies. In essence, each secured object has an ACL that consists of data from a column of the access control matrix. Only entries for subjects allowed to use the object are present in the ACL thereby eliminating the access control matrix's inefficiencies. Replacing the ACL for an object is easy, whereas determining all of the privileges for a single subject is difficult in such a system [SAN94]. For instance, if all of the access privileges of a subject must be revoked, then all ACLs must be examined.

2.2.3 Capabilities

Another form of implementation of the access control matrix is the capability model where each subject possesses a list that indicates what operations on which objects the subject is authorized to perform. A subject's capabilities relate to its row in the access control matrix. The capabilities list structure allows each subject's privileges to be easily examined, while determining which subjects can access a particular object becomes a difficult task [SAN94].

2.2.4 Authorization Relations

In the authorization relation, the access control matrix is represented as a table of tuples of the form <subject, access mode, object> where each tuple specifies an access right of one subject on one object. If the table is sorted by subject, it acts much like capabilities lists, whereas if it is sorted by object, it acts much like a list of ACLs. Such a representation is often found in relational database systems [SAN94].

2.2.5 Discretionary Access Control (DAC)

In DAC based systems, the subject's identity, object identities, and a series of rules represent the policy that is consulted to determine if a particular subject is allowed to a perform a specific operation on an object. Closed DAC systems only allow accesses when the policy specifically allows the access while an open DAC system only disallows access when the policy denies access. Some systems allow for a combination of positive and negative authorizations allowing specification of both allowed and forbidden accesses. One weakness of DAC is that once a user gains access to information, there is potentially nothing stopping that user from sharing the information with another user who may not be allowed to access it [SAN94]. In addition, in DAC systems it can be difficult to represent and maintain complex access control policies for a large organization.

2.2.6 Mandatory Access Control (MAC)

In a MAC based system, objects are given a security level (i.e., Top Secret, Secret, Confidential, and Unclassified) and each is subject assigned a clearance level (i.e., Top Secret, Secret, Confidential, and Unclassified) reflecting the subject's trustworthiness. Access to an object is granted to a subject only if the subject's clearance level is sufficiently high to access the object's security level. A subject may read objects of equal or lower security level than the subject's clearance level, whereas the subject may only write to objects that are of equal or higher security level than the subject's clearance level. This structure makes sure that information from an object with a higher security level does not get written to a document with a lower security level and create a breach in security [SAN94].

2.2.7 Role-Based Access Control (RBAC)

Role-based access control is a variation on DAC where there is an additional level of indirection between subjects and their privileges [SAU01]. In general, users are assigned to one or more roles, and roles are linked to access control privileges. In such a system, roles can be designed to match an organization's structure and user's roles can be easily changed as their position in an organization changes. This gives RBAC a level of flexibility in management that DAC does not possess. There are sophisticated variations on RBAC that introduce hierarchical relationships between roles as well as between permissions and roles and between users and roles. As an example, two roles can be defined as mutually exclusive so that a single user cannot hold both roles [SAN96].

Roles can also have inheritance relations so that one role can inherit permission assigned to another role [ANS03].

These access control models are different ways of representing the relationships between the authorization database and reference monitor. The goal of this thesis is to develop an access control framework for UniFrame that allows a variety of types of access control schemes from which to choose when creating a system. Therefore it is not necessary to decide which of the previous models is most appropriate for UniFrame in general, but instead, leave as much of that decision as possible up to the system integrator's judgment when the system is created. Therefore, the UniFrame access control model must attempt to be independent of the type of policy mechanism used.

2.3 Model Driven Access Control

One way of modeling access control independent of specific technologies or policy models, is to develop a more general access control model that can be refined into each of the major existing access control models. [BUR03] makes progress along these lines by proposing a platform independent model for access control which can inform the development of access control portion of the UniFrame knowledge-base as well as UMM specifications for components. OMG's Model Driven Architecture (MDA) facilitates the creation of standard Platform Independent Models (PIMs) and the means for transforming PIMs into Platform Specific Models (PSMs) for implementation. By creating a PIM of access control that can accommodate currently existing access control models, the abstract view of access control can be used at the modeling level for parameterization of domain models. This will enable the transformation of models to platform specific models incorporating access control points.

[BUR03] Burt identifies five principles that guide the development of Model Driven Access Control:

1. "Access Control Points should be identifiable via parameterization of the domain PIMs with a single model element."
2. "Protected Resources should be identifiable via parameterization of the domain PIMs with a single model element."

3. "Access policy should be defined, developed and managed separately from the application business logic. Access policy rules and access policy models must be able to evolve without modification of the business software."
4. "The policy model (role-based, user-based, code-based…) should not be exposed to the business application developers. That is, the business logic should have absolutely no knowledge of the access control model utilized to make access decisions."
5. "The access control platform independent model must provide abstractions that are consistent with existing commonly used platform specific models for access control."

As per [BUR03], there are three pieces of information that must be provided during the parameterization of the domain model: identification of the resources that might require protection, identification of the points within the application where the access control checks should be made, and identification of the application specific context/attribute information that might be needed at the point where an access control check is made. These three pieces of information are represented as the following modeling elements: *ResourceId* represents the resource and manages its identity, *Guards* represent the access control points in the application, and *DynamicContextServer* represents context information used in making access decisions.

The *ResourceId* can represent any physical, logical, or conceptual resource or even a group of resources. A *Guard*, typically a software piece protecting an electronic resource, is inserted in the application at every point where an access control check is to be enforced. Whereas *ResourceIds* and *Guards* will always be required when parameterizing the domain model, the *DynamicContextServer* will only be needed in some sophisticated applications that require the ability to plug-in custom context servers. The access control guard enforces access control decisions based on advice from the *AccessManager,* which in turn accesses one or more attribute server(s) and one or more *AuthorizationService(s)*.

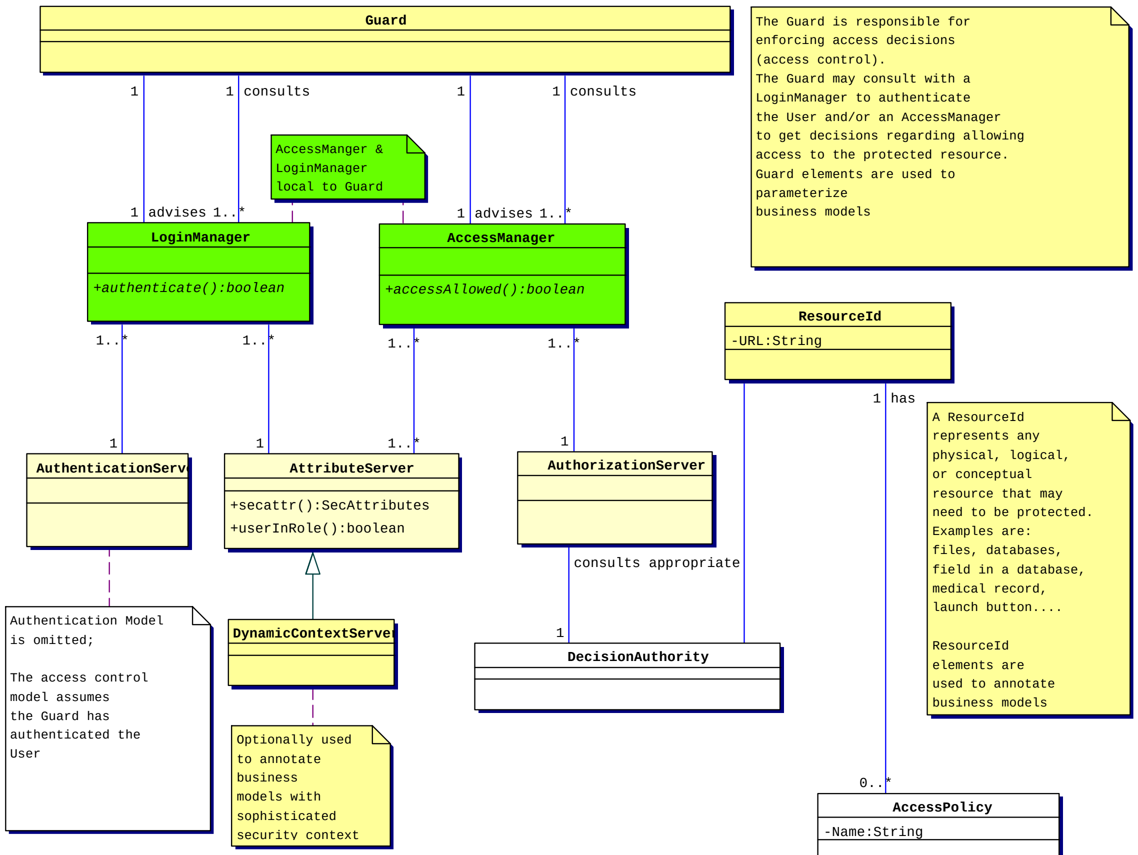
Guard
The Guard is responsible for enforcing access decisions (access control). The Guard may consult with a LoginManager to authenticate the User and/or an AccessManager to get decisions regarding allowing access to the protected resource. Guard elements are used to parameterize business models
1
1 consults
1
1 consults
AccessManger & LoginManager local to Guard
1 advises 1..*
1 advises 1..*
LoginManager
+authenticate():boolean
AccessManager
+accessAllowed():boolean
ResourceId
-URL:String
1..*
1..*
1..*
1..*
1 has
A ResourceId represents any physical, logical, or conceptual resource that may need to be protected. Examples are: files, databases, field in a database, medical record, launch button....
ResourceId elements are used to annotate business models
1
1
1..*
1
AuthenticationServ
AttributeServer
+secattr():SecAttributes
+userInRole():boolean
AuthorizationServer
consults appropriate
Authentication Model is omitted;
The access control model assumes the Guard has authenticated the User
DynamicContextServe
1
DecisionAuthority
Optionally used to annotate business models with sophisticated security context
0..*
AccessPolicy
-Name:String

Figure 2-2: Platform Independent Model for Access Control [BUR03]

It is clear from [BUR03]'s model driven access control that the knowledge-base in UniFrame must have a means of representing resources in a domain model. This provides a standardized vocabulary for *ResourceIds* for the knowledge-base which will be crucial for writing system specifications and for searching for components in the URDS. Identifying which resources a component protects with *Guards* is necessary in order to search for components that can protect resources specified by the system integrator. Therefore, the model driven access control suggests that resources and guards are the key parts of the access control PIM that characterize the access control properties of an implemented component.

2.4 Common Criteria

In addition to model driven access control, the Common Criteria (CC) also provides insight characterizing access control, but in the context of additional security properties. In an effort to develop an international standard for evaluating the security characteristics of information technology products, the CC represents an alignment of preexisting US, Canadian, and European standards [CC04]. It is hoped that this new standard will result in the world wide mutual recognition of evaluation results for IT products. Common Criteria requirements are broken down into two categories: functional requirements that specify security behavior and assurance requirements which are the basis for gaining confidence that the desired security properties of a product are effective and implemented correctly.

To organize these requirements into logical packages for individual classes of products, CC defines the Protection Profile (PP) construct in order to allow consumers and developers to create standardized sets of security requirements that meet their needs. The Target of Evalution (TOE) is the portion of the product subject to evaluation and the TOE's security threats, objectives, and requirements are summarized in the Security Target (ST) document for use as a basis for evaluation of a specific product. The TOE is evaluated by an examination of the Security Target, a set of evidence about the TOE, and the TOE itself. The Protection Profile contains an implementation-independent set of security requirements and objectives for a class of products and is intended for reuse, while the Security Target is a set of requirements and objectives for a specific TOE implementation. A Security Target can claim conformance to more than one Protection Profile.

The Protection Profile and Security Target documents are based on carefully structured sets of functional and assurance security requirements outlined by the Common Criteria. Criteria are first organized into classes representing the most general level of grouping, with members of the same class sharing a common focus.

Table 2-1: Common Criteria Functional Security Classes

Class FAU: Security audit
Class FCO: Communication
Class FCS: Cryptographic support
Class FDP: User data protection
Class FIA: Identification and authentication
Class FMT: Security Management
Class FPR: Privacy
Class FPT: Protection of the TSF
Class FRU: Resource utilization
Class FTA: TOE access
Class FTP: Trusted path/channels

Each class is composed of one or more families with each family being a group of requirements that share the same objective but may differ in rigor or emphasis. Each family is composed of one or more components and a component contains a defined set of security requirements and represents the smallest selectable set of security requirements for inclusion in a PP or ST. Components within a family may be ordered to represent increasing strength or capability of requirements that have a common purpose. Dependencies may arise between components when one component is not self sufficient and relies upon another component.

Components can be used exactly as specified by the CC or they can be modified through permitted operations in order to meet specific security objectives. There are four such permitted operations: iteration, assignment, selection and refinement. Each component will define the types of modification operations that are permitted. The operations of iteration and refinement can be performed on any component.

The User Data Protection class is split into thirteen families that address user data within a TOE, during import, export, and storage as well as security attributes directly related to user data. Of these families, two families that relate directly to the scope of this thesis: FDP_ACC:Access control policy, and FDP_ACF:Access control functions. Table

2-1 and Table 2-2 summarize the specifications for these two families as found in the Common Criteria documentation [CC04]. FDP_ACC deals with specifying the overall type of access control system (DAC, MAC, RBAC, etc.) along with the subjects, objects, and operations on the objects that are in the scope of the access control policy. FDP_ACF deals with the details of how access control decisions are carried out.

Table 2-2: Class FDP: User Data Protection Families

(FDP_ACC) Access control policy
(FDP_ACF) Access control functions
(FDP_DAU) Data authentication
(FDP_ETC) Export to outside TSF control
(FDP_IFC) Information flow control policy
(FDP_IFF) Information flow control functions
(FDP_ITC) Import from outside TSF control
(FDP_ITT) Internal TOE transfer
(FDP_RIP) Residual information protection
(FDP_SDI) Rollback (FDP_ROL) Stored data integrity
(FDP_UCT) Inter-TSF user data confidentiality transfer protection
(FDP_UIT) Inter-TSF user data integrity transfer protection

In agreement with [BUR03]'s model driven access control, the FDP_ACC family requires the naming of objects under the control of the access control functionality. This reinforces the claim that named resources are key elements of both the domain model as well as component specifications. Other information contained in the FDP_ACC and FDP_ACF families relate to subjects and policy models that, though necessary for a complete access control system, need not play a role in characterizing software components since the goal is to not have a component being policy agnostic. Therefore, the Common Criteria reinforces the notion of naming resources in the domain model for a system. In addition, the FDP_ACC family suggests that a component needs to specify through which operations a resource will be accessed. This thesis addresses how resource and operations are to be represented in a component's specifications.

Table 2-3: Access Control Policy (FDP_ACC) Family [CC04]

Family Behavior: This family identifies the access control security function policies (SFP) (by name) and defines the scope of control of the policies that form the identified access control portion of the TSP. This scope of control is characterized by three sets: the subjects under control of the policy, the objects under control of the policy, and the operations among controlled subjects and controlled objects that are covered by the policy. The criteria allows multiple policies to exist, each having a unique name. This is accomplished by iterating components from this family once for each named access control policy. The rules that define the functionality of an access control Security Function Policy will be defined by other families such as FDP_ACF and FDP_SDI. The names of the access control SFPs identified here in FDP_ACC are meant to be used throughout the remainder of the functional components that have an operation that calls for an assignment or selection of an "access control SFP."

Component leveling:

FDP_ACC.1 Subset access control requires that each identified access control SFP be in place for a subset of the possible operations on a subset of the objects in the TOE.

> FDP_ACC.1.1 The TSF shall enforce the [assignment: access control SFP] on [assignment: list of subjects, objects, and operations among subjects and objects covered by the SFP].

FDP_ACC.2 Complete access control requires that each identified access control SFP cover all operations on subjects and objects covered by that SFP. It further requires that all objects and operations with the TSC are covered by at least one identified access control SFP.

> FDP_ACC.2.1 The TSF shall enforce the [assignment: access control SFP] on [assignment: list of subjects and objects] and all operations among subjects and objects covered by the SFP.
>
> FDP_ACC.2.2 The TSF shall ensure that all operations between any subject in the TSC and any object within the TSC are covered by an access control SFP.

The Common Criteria does not address the composition of security properties. According to [KAT02], there is no formal/mathematical approach for determining the security properties of a system composed of evaluated products. Therefore, CC does not provide any direct guidance for developing at means of composing access control specification of components into the specifications of a system.

Table 2-4: Access Control Functions (FDP_ACF) Family [CC04]

Family Behavior: This family describes the rules for the specific functions that can implement an access control policy named in FDP_ACC. FDP_ACC specifies the scope of control of the policy.

Component leveling: This family addresses security attribute usage and characteristics of policies. The component within this family is meant to be used to describe the rules for the function that implements the SFP as identified in FDP_ACC. The PP/ST author may also iterate this component to address multiple policies in the TOE.

FDP_ACF.1 Security attribute based access control allows the TSF to enforce access based upon security attributes and named groups of attributes. Furthermore, the TSF may have the ability to explicitly authorize or deny access to an object based upon security attributes.

FDP_ACF.1.1 The TSF shall enforce the [assignment: access control SFP] to objects based on [assignment: security attributes, named groups of security attributes].

FDP_ACF.1.2 The TSF shall enforce the following rules to determine if an operation among controlled subjects and controlled objects is allowed: [assignment: rules Governing access among controlled subjects and controlled objects using controlled operations on controlled objects].

FDP_ACF.1.3 The TSF shall explicitly authorize access of subjects to objects based on the following additional rules: [assignment: rules, based on security attributes, that explicitly authorize access of subjects to objects].

FDP_ACF.1.4 The TSF shall explicitly deny access of subjects to objects based on the [assignment: rules, based on security attributes, that explicitly deny access of subjects to objects].

2.5 Access Control in DCS

Beyond the variety of access control models, researchers have also explored different methods for incorporating access control mechanisms into distributed computing systems. Each model is discussed along with advantages and disadvantages, which will be considered while developing the UniFrame access control model.

2.5.1 Federated System Models

The implementation of RBAC for federated information systems on the world wide web discussed in [TAY03] aims to provide a structure for creating federations of distributed heterogeneous databases. This security model supports single sign-on authentication, role-based access control, high levels of autonomy for local database managers, and a low maintenance overhead. Taylor and Murty propose a three level architecture consisting of client software such as web browsers or Java applets, which interact with midlevel brokers that provide an integrated interface to multiple databases comprising the lowest level. User authentication would take place at the broker while access control decisions are all carried out at the database level. This model concerns itself with only securing read access and does not address scalability of the system beyond the three layers proposed [TAY03]. An access control model of general DCS should attempt to address a variety of potential modes of access to the system resources and should address a range of possible architectures.

In their article on access control in federated systems [VIM97], Vimercati and Samarati proposed a two level structure for data. Federated objects are stored at the global level in a central site with authorization specified in terms of federated users and groups. At the local level, administrators may export local data to the federation which entails allowing a reference to the data to be stored in the federation. Accessing local data referenced in the federation, the federation communicates with the appropriate site for both authorization and data retrieval since that data is not actually stored in the federation. Relying upon a centralized federation space for storing both federation objects and references to local objects, this access control model may have scalability problems for federations with many local sites and a large amount of data and, therefore, may not be flexible enough to be used in a variety of DCS.

2.5.2 Law Governed Architecture

Minksy and Ungureanu [MIN98] have implemented a prototype system based on the concept of "law-governed architecture" and "regulated interactions" (RI) that can support a wide range of security models and policies ranging from discretionary access control and mandatory access control to Chinese firewall policy and sealed-bid auction policies. They define a security policy *P* to be the triplet *<M,G,L>* where *M* is the set of messages regulated by *P*, *G* is group of distributed agents, and *L* is the set of rules regulating the exchange of messages. In general, *L* determines who in *G* can send which messages to whom and what the effect of such messages should be.

An agent, which may be a client or server, interacts with the environment through a controller that regulates the flow of messages. CS represents the control state of the agent and is the part of the law that is associated with that particular agent. R represents the global set of rules for any regulated event that happens at an agent with a given control state. In addition, all controllers must have identical copies of the law, and the law, universal system-wide rules, is represented by a prolog program in the case of the author's system [MIN98]. A centralized server referred to as the secretary, maintains a persistent copy of the Law and is responsible for distributing the Law and for maintaining the control states of all group members.

This model assumes that messages between agents provide the necessary granularity of access control in the system. Systems needing particularly fine grained access control may need complex rules and controller states to properly manage accesses. It is also easy to envisage situations in which an agent's use of its internal resources may change over time rendering the controller's state out of sync with the agents services especially considering that components have private implementations. In addition, fault tolerance and scalability of this model are limited by the centralized secretary. The law-governed architecture does have the advantage of allowing the system integrator to add reference monitors to act as wrappers around components, thereby providing the potential for adding addition access control measures to that system than are provided by the components alone.

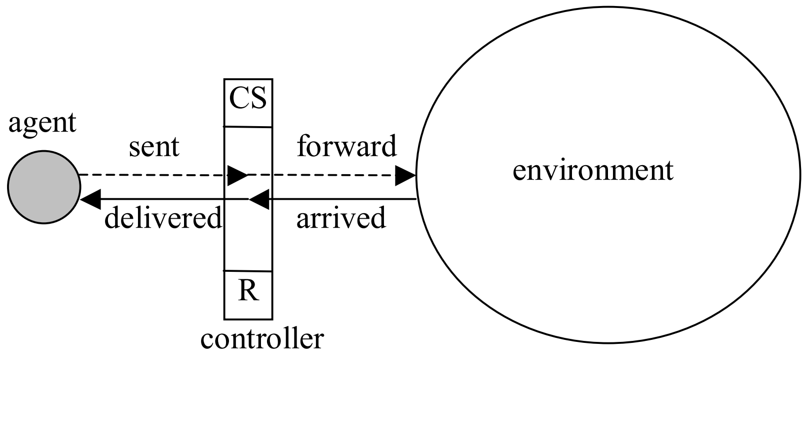

Figure 2-3: Architecture for Enforcement of the Law

2.5.3 Active Capabilities Framework

[ABE03] extends the idea of capabilities to create what is called active software capabilities framework (ASCap) that centers around the idea of a policy object, which instead of being embedded into the access control component, is delivered by the client. The client firsts requests an ASCap (policy object) from the security server. Then the client and the object server both must instantiate proxies. The client ASCap proxy may then request additional credentials from other servers which may identify roles or other security attributes. The client then can invoke the server through the client proxy. During this invocation, the client proxy adds the policy object and any additional credentials to the request and forwards the request to the server. The server then invokes the policy object in order to determine if the request is authorized. Abendroth and Jensen [ABE03] have used this framework to implement and evaluate both discretionary access control and role-based access control setups. They conclude that the ASCap implementation does have a 50% increase in response time as seen by the client mainly due to the transmission over the policy object and credentials with each request. Because the server ASCap proxy can be intimately connected with the server component, it should be possible to use this framework for fine grained access control. It would have an advantage over a centralized policy by off loading this work onto the clients. Though these advantages may be outweighed by the performance problems cited previously.

2.5.4 Cryptographic Based Access Control

Harrington and Jensen [HAR03] propose a cryptography-based access control paradigm for distributed file systems in which reference monitors are no longer needed to guard resources. All protected data is stored in encrypted form using asymmetric encryption with data being encrypted with a private key. This implicitly allows only principals in possession of the public key the privilege of reading the data. Write protection is accomplished through using the private key encryption along with a journaling mechanism to maintain a history of the state of data in the system. If data is written to the system with an improper private key, when a principal then attempts to read the data with the correct public key, the system will identify the mismatch in keys and revert to the next older version of the data stored in the journaling system.

The performance evaluation of an implementation of this model indicates that read performance improved by 20% over a centralized access control model. The write access performance degraded by two orders of magnitude with a large standard deviation, indicating very large variability of the achieved write durations [HAR03]. Such slow write performance would indicate that this model would only be useful in a system that performs mainly read operations with very few writes. In addition, changing a principal’s access privileges would require changing a datum’s key pairs and propagating the new keys to authorized users. This may also require decrypting and re-encrypting all existing instances of the data in the journaling system to make sure a removed principal cannot access any data. Otherwise such a user would still possess the key necessary for decrypting the data. These weaknesses indicate that the cryptography based access control approach would not have wide applicability in a variety of DCS.

2.5.5 Calculus of Distributed Access Control

Abadi [ABA93] created the concepts, protocols and algorithms for addressing access control in distributed systems. Much of the model centers on the problem of identifying the principal who is attempting to carry out an operation in the DCS. "Who is speaking?" and "who is trusted?" in a DCS can be complex to determine and Abadi extends the idea of a principal to include the following:

- Users and machines
- Channels – input devices, cryptographic channels
- Conjunctions of principals – *A and B say s*
- Groups
- Principals in Roles – *A as R*
- Principals on behalf of principals – *B for A*

Abadi develops a calculus of principals that is applied through access control lists (ACL) to determine if a particular composite principal may access a secured object. For instance, assuming that component C knows B's key,

K_B => B,

if C sees a message requesting r on behalf of A,

K_B says A says r,

then C can determine

B says A says r.

C then consults the ACL for *r* to see if the composite principal *B says A* is trusted on *r*. Abadi determines that this general access control model requires exponential time. Through a series of restrictions on the calculus aimed at limiting complexity while maintaining flexibility, he describes a prototype application as having adequate speed suggesting that the algorithm may be practical.

Though such a framework could provide UniFrame systems with a formal basis for determining identity of principals in the system, this thesis assumes the existence of an authentication and identification scheme for principals. In addition, the thesis does not concern itself with specific access control policy evaluators as proposed by Abadi, but instead proposes a framework that can accommodate any policy evaluator required by the system integrator.

Several insights from these distributed models that can inform the development of the access control framework for UniFrame systems:

1. The access control framework must support the efficient protection of all potential types of operation on a resource (ie. read, write, execute, etc.).
2. The framework should provide for scalability to accommodate a potentially large system and distribute access control evaluation through the system.
3. Avoid the frequent transmission of large amounts of access control information throughout the system.
4. Provide for fine grained access control.
5. Provide a mechanism for tracking the identity of "who" is performing any given action in the system.

It will be assumed that the fifth point will be implemented using a mechanism such as Abadi's calculus of distributed access control. The first four points will be directly addressed in this thesis.

2.6 Composing Access Control Policies

Most access control systems assume a monolithic and complete access control policy. In some cases, this is difficult to achieve when several independent security domains must collaborate in a DCS. To address this issue, [BON00] has developed a modular approach for composing access control policies into a unified coherent policy through the use of an algebra of policies. Through the use of operators such as addition, union, subtraction, closure, scoping restriction, overriding, and template, Bonatti develops the means of systematically combining heterogeneous access control policies. For instance, scoping restriction allows for restricting the application of a policy to a given set of subjects, objects, and actions. In addition, the authors claim that the formal semantics of the algebra allow verification of correctness requirements for the composed policy [BON00]. Through the use of these techniques it is possible to express in logic the complete composed policy. In the cases when a policy cannot be known, it is also possible to maintain run-time links in the composed policy to policies that are "black boxes."

Though Bonatti's work can prove useful in DCS with different entities contributing portions of the access control policy, this approach does not have a direct impact on the contributions of this thesis. It is assumed in this thesis that components are to be made available with as much flexibility in configuration as possible. This goal would indicate that individual components should not come with preconfigured policies. Instead the system integrator will supply the access control policy for the final composed system. For instance, the power of RBAC is in the ability of a organization to represent its personnel structure as roles in the access control system. If a component were to come with a RBAC based policy, then the roles in this policy may not map easily to the company's roles. Or perhaps a component comes with a DAC-based policy and the system requirements are for a RBAC-based policy. It would be better for components not to include predefined policies in order to maintain the broadest applicability in distributed systems. By being policy agnostic, UniFrame can create a market place for companies to create and sell access control policies for different application domains.

2.7 Composing Security Properties

In their efforts to characterize the security properties of software components, Khan and Han [KHA02] [KHA03] propose a model based on components expressing their required and ensured security properties. Requirements are expressed as predicates and a compositional security contract is created for each composition of two components, X and Y, by checking to see if the ensured security properties of X imply the required properties of Y and visa versa.

$$CSC_{X,Y} = (E_X \Rightarrow R_Y) \wedge (E_Y \Rightarrow R_X)$$

If this expression evaluates to true, then there is a trustable security composition between the two components. Ensured properties are represented as facts in logic programming while required properties are represented by headed horn clauses. By combining the logic statements from two components, it is possible in this framework [KHA02] to determine if a valid compositional security contract can be derived. This approach will be applied in this thesis to the matching algorithm in the URDS to search for components with particular access control mechanisms because it has the potential to be general enough to handle additional security properties beyond access control, such as encryption [KHA03].

2.8 Formal Modeling

To achieve a high degree of assurance as to the behavior of a computing system, it may be necessary to formally model the system in order to verify its characteristics before implementation. There are a variety of means of formal modeling, two of which are discussed in detail below for consideration for use in verification of the access control behavior of a system.

2.8.1 Petri Nets

Petri nets are a mathematical tool for mathematically modeling systems. Originally developed by Carl Adam Petri [PET62], Petri nets are directed, weighted, bipartite graphs with two types of nodes: places and transitions. The edges in the graph go either from a place to a transition or from a transition to a place with places represented graphically as circles and transitions represented as boxes or bars. The edges are labeled with an integer weight with unlabeled edges assumed to have a weight of one. Each place is assigned a non-negative integer value indicating its state and identifying the number of tokens held by that place with tokens represented by black dots. Each transition has a certain number of input and output places referred to as preconditions and postconditions respectively and the transition itself is often called an event.

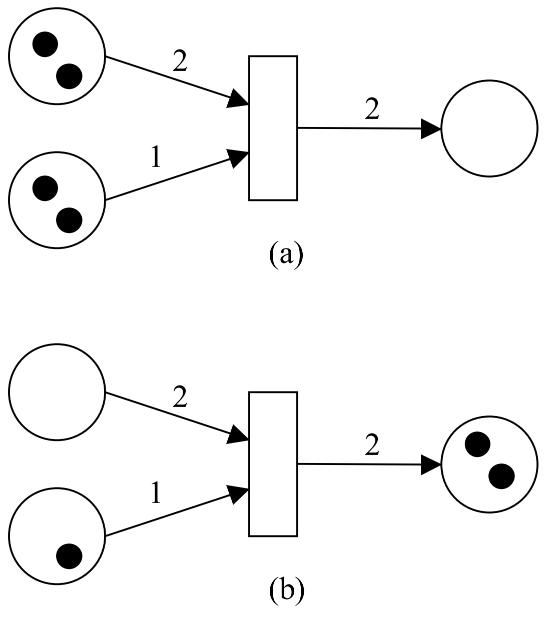

Figure 2-4: Example Petri Net: (a) Before the Transition, (b) After the Transition

The state of the system is represented by the distribution of tokens among all of the places in the model and system behavior is resulting from the following three rules determines how the model's state may evolve with time:

1. A transition is *enabled* if each input place of the transition is marked with at least as many tokens as the weight of the arc connecting the input place to the transition.
2. An enabled transition may or may not fire depending on whether or not the event takes place.
3. Firing of a transition results in removing the number of tokens from the input place indicated by the weight of the arc and adding the number of tokens to the

output place indicated by the weight of the arc from the transition to the output place [MUR89].

Figure 2-4 provides an example Petri net with (a) representing the state before transition and (b) the state after transition.

According to Murata, Petri nets have been used to model dataflow computation, communications protocols, synchronization control, formal languages, and even multiprocessor systems. Through the use of Petri nets, several behavioral properties of the model can be deduced: reachability, boundedness, liveness, reversibility, coverability, persistence,and fairness with each being defined in [MUR89]. Colored Petri nets (CPN) are an extension of Petri nets that labels the tokens in each place with a type referred to as color. In addition, the edges in colored Petri nets can have conditional statements for determining their weight. Such extensions allow for more concise modeling of more complex systems [KRI98]. To handle large colored Petri nets, hierarchical colored Petri nets were developed so that Petri nets could be modularized [KRI98].

2.8.2 Temporal Logic of Actions

According to Leslie Lamport [LAM94], a concurrent algorithm is usually specified with a program, and the correctness of the algorithm means that the program satisfies certain properties. Temporal Logic of Actions [TLA] was created in the late 1980’s by Lamport to express algorithms and their properties as logic formulas. In this way, the correctness of an algorithm means that the formulas specifying the algorithm imply the formulas specifying the properties. TLA was refined over many years into the more current version referred to as TLA+ [LAM03].

TLA specifications can be broken down into two areas: safety properties, which require almost no temporal logic, and liveness properties where temporal logic plays a central role. In general, the specification of a system can be reduced mathematically to one TLA+ formula.

$$\text{Spec} \overset{\Delta}{=} \text{Init} \wedge \Box[\text{Next}]_{\text{variables}} \wedge \text{Liveness}$$

Init represents an initial predicate that specifies the possible initial values of all variables in the system. *Next* is a predicate that expresses the relation between the current state of all variables and the next state(s) of the system. The temporal operator □ (pronounced box) asserts that a formula is always true. Therefore $\Box[\mathit{Next}]_{\mathit{variables}}$ states that *Next* is true for every step in the system's behavior. Given this, $\mathit{Init} \wedge \Box[\mathit{Next}]_{\mathit{variables}}$ is true of a behavior iff the initial state satisfies *Init* and every step satisfies *Next.* Specifications are written as predicates with different predicates evaluating to true in different states. These predicates are similar to transitions in Petri nets, though they can incorporate more complex logic. The *Spec* expression also includes a conjunct with *Liveness* which represents an expression of what must happen in the model. Given a specification, it is possible to evaluate safety and liveness properties through using a model checker to explore the state space [LAM03].

Though Petri nets and TLA+ share much in common, in the context of this thesis it was found that TLA+ could more concisely represent access control behaviors of a system and it has a freely available and modifiable java-based model checker. In addition, TLA provides a straightforward means for expressing logical properties for the model checker to verify. For these reasons TLA+ was chosen as the method for dynamically modeling in this study. Chapter 7 will explain the details of TLA+ that are relevant to the modeling used in this thesis.

This chapter provides an overview of previous and related work that has guided the ideas presented in this thesis. The next chapter will set the stage for the case study used to validate ideas developed in Chapters 4, 5, and 6. The case study will be fully developed in Chapter 7.

3. INTRODUCTION TO THE CASE STUDY

The ideas proposed in this thesis are presented in the context of a case study based on a simplified university student information system. This case study was chosen because of the need for a university to store sensitive student information and share this information with a variety of different stakeholders in the university community. A complete UniFrame Generative Domain Model (UGDM) as presented in [HUA03] will not be developed. Instead, a specific instance of a system will be used to demonstrate how the ideas developed in this thesis are implemented in the UniFrame approach. Section 3.1 discusses issues to be addressed by a case study in order to provide realistic testing results. Section 3.2 provides an overview of the functionality for the student information system. Section 3.3 discusses the system architecture and Section 3.4 details the patterns of interaction between the system components. This simplified student information system model will serve as case study throughout this thesis and each of the following chapters will augment portions of this model to illustrate how the UniFrame access control model is constructed and verified.

3.1 Access Control Issues to Be Addressed

There are several issues that must be directly addressed in order to create a case study that effectively tests the access control modeling proposed in this thesis. Based on the background research, access control behavior can be categorized into two groups. First, there are situations where guards are used to filter information in the system being accessed by the user. For instance, any user can request to read any student record but they will only be allowed to see certain sections of each student's record. The read command is not blocked, but the information released to the user is limited. Second, there are situations in which the user's access to a resource is denied by a guard, and the request cannot be completed. If a user tries to execute a system command that requires writing to a file and the system denies the user access to the file, then the command must completely fail. The case study must represent each of these two situations. Beyond these two types of behaviors, the case study must also address dynamic security attributes that are added to a user's attribute list when an access control check is made. The case study must demonstrate a means of using such dynamic attribute properly. The case study should also contain a variety of resources so that component searching can be demonstrated using a variety of system specifications. In addition, multiple variations of each component type should be developed to demonstrate the ability of this approach to properly represent variations in component implementation. Each of these issues is addressed in the case study used in this thesis.

3.2 Basic System Functionality

The student information system case study provides the basic functionality used to store and retrieve academic and financial information for students enrolled at a university. The goal is not to model a comprehensive student information system, but instead to use representative functionality from the system to provide a realistic setting in which to test the access control model presented in this thesis. Therefore, the system functionality will be limited to the following actions:

- Create a new student record
- Read\Save student records

The read student record action will attempt to read the information for one student from each of the servers in the system. From the aspect of access control, a user may only be able to read portions of the student record, and therefore the access control infrastructure acts much like a filter for information in this situation. The save student record action will attempt to write portions of a student's record to various servers in the system. The save action will only succeed if each portion of the record being written is successfully stored. In this case, the access control infrastructure acts much like a gate that either allows the entire action or cancels the action al together.

3.3 System Architecture

To implement the basic functionality described previously, the student information system instance used in this case study consists of five separate components: User Terminal, Record Server, Academic Server, Financial Aid Server, and the Student Employment Server. Figure 3-1 graphically represents how these components are composed into the student information system as a whole. The functionality of each component in the system, summarized in Table 3-1, distributes the storage of various sections of the student record across multiple servers. User requests are submitted via the User Terminal component and are all routed through the record server before being sent to one or more lower level servers. This architecture will create a variety of situations in which one action taken by a user will require access control verifications across multiple distributed components.

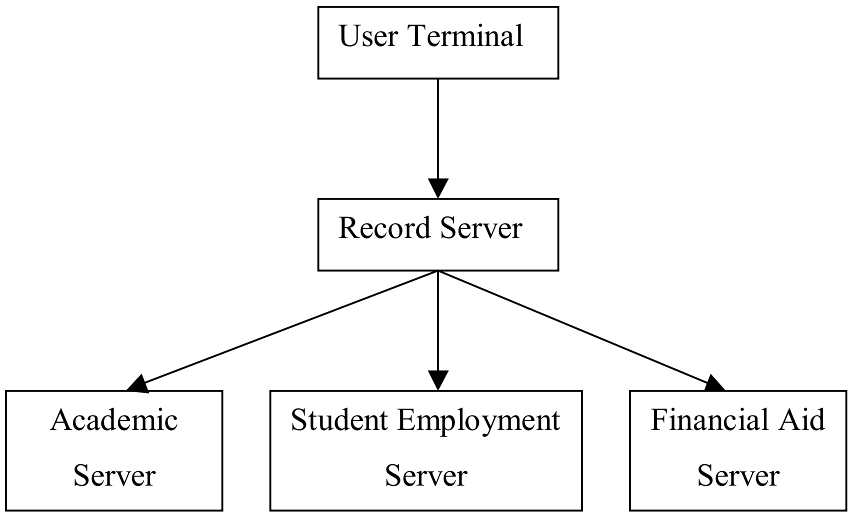

Figure 3-1: Student Information System

Table 3-1: Student Information System Components

Component	Purpose
UserTerminal: (UT)	Provides a uniform access point for all users across the system.
RecordServer: (RS)	Acts as the main contact point for all requests. It holds general information about all students in the system.
AcademicServer: (AS)	Holds transcripts and degree program information for each student.
FinancialAidServer: (FAS)	Holds student loan, grant, and scholarship information.
StudentEmploymentServer: (SES)	Holds student employment information for students working at the university.

Each component will support one or more remote interfaces used to knit together the system. The different interfaces and associated method signatures are shown in Table 3-2. In accordance with the UGDM format, each abstract component type will define the interfaces to be provided by the component and the interfaces required on post-processing collaborators in the system. This information is defined in the UMM specifications for each abstract component type in the system and the UMM specifications for the five abstract components in the student information system are given in Appendix A. As a summary, Table 3-3 shows how these interfaces are distributed across the different components in the system and Table 3-4 presents the abstract component specifications for the record server component.

Table 3-2: Remote Interfaces for the Student Information System

Interface	Method
IStudentManagement	addStudent(String sid)
IStudentRecord	StudentRecord readStudentRecord(String sid) saveStudentRecord(String sid, StudentRecord sr, int tid)
IFinancialAid	FinancialAid readFinancialAid(String sid) saveFinancialAid(String sid, FinancialAid fa, TransactionId tid)
IStudentEmployment	StudentEmployment readStudentEmployment(String sid) saveStudentEmployment(String sid, StudentEmployment se, TransactionId tid)
IAcademicRecord	AcademicRecord readAcademicRecord(String sid) saveAcademicRecord(String sid, AcademicRecord ar, TransactionId tid)

ITransaction	TransationId transaction()
	commit(int tid)
	rollback(int tid)

Table 3-3: Component Interfaces

Component	Interface(s)
UserTerminal: (UT)	IStudentRecord, IStudentManagment
RecordServer: (RS)	IStudentRecord, IStudentManagement
AcademicServer: (AS)	IAcademicRecord, IStudentManagement, ITransaction
FinancialAidServer: (FAS)	IFinancialAid, IStudentManagement, ITransaction
StudentEmploymentServer: (SES)	IStudentEmployment, IStudentManagement, ITransaction

Table 3-4: Abstract Component Specification for the Record Server

Abstract Component: *RecordServerCase1*

1. Component Name: *RecordServer1*
2. Component Subcase: *RecordServerCase1*
3. Domain Name: University
4. System Name: Student Information
5. Informal Description: Provide a basic student information system for a university.
6. Computational Attributes:
    - 6.1 Inherent Attributes:
        - 6.1.1 id: N/A
        - 6.1.2 Version: version 1.0
        - 6.1.3 Author: N/A
        - 6.1.4 Date: N/A
        - 6.1.5 Validity: N/A
        - 6.1.6 Atomicity: Yes
        - 6.1.7 Registration: N/A
        - 6.1.8 Model: N/A
    - 6.2 Functional Attributes:
        - 6.2.1 Function description: store general student information and coordinate requests to additional servers.
        - 6.2.2 Algorithm: N/A
        - 6.2.3 Complexity: N/A
        - 6.2.4 Syntactic Contract
            - 6.2.4.1 Provided Interface: *1StudentManagement*, *IStudentRecord*
            - 6.2.4.2 Required Interface: *IStudentManagement, IStudentRecord, IAcademicRecord, IStudentEmployment, IFinancialAid, ITransaction*
        - 6.2.5 Technology: N/A
        - 6.2.6 Expected Resources: N/A
        - 6.2.7 Design Patterns: NONE
        - 6.2.8 Known Usage: NONE
        - 6.2.9 Alias: NONE
7. Cooperation Attributes
    - 7.1 Preprocessing Collaborators:*UserTerminalCase1*
    - 7.2 Postprocessing Collaborators: *AcademicServerCase1, FinancialAidServerCase1, StudentEmploymentServerCase1*
8. Auxiliary Attributes:
    - 8.1 Mobility: No
    - 8.2 Security
    - 8.3 Fault tolerance: *L0*
9. Quality of Service
    - 9.1 QoS Metrics: *throughput*, *end-to-end delay*
    - 9.2 QoS Level: N/A
    - 9.3 Cost: N/A
    - 9.4 Quality Level: N/A

3.4 Component Interactions

Beyond the static description of architecture and component characteristics, it is also necessary to describe how the five components will interact with each other as system functionality is carried out. There are three specific commands that a user may submit to the system that are to be modeled: add student, read student record, and save student record. The add user is necessary for putting students into the system and simply creates an empty record for a student across all servers in the system. In each of the read, save, and add actions, processes at each of the four servers can occur concurrently or sequentially depending on the record server design. Because the success or failure of writing of a student record depends on the success or failure of multiple components, the system uses transactions to commit or rollback the save operation across all components.

Read Student Record

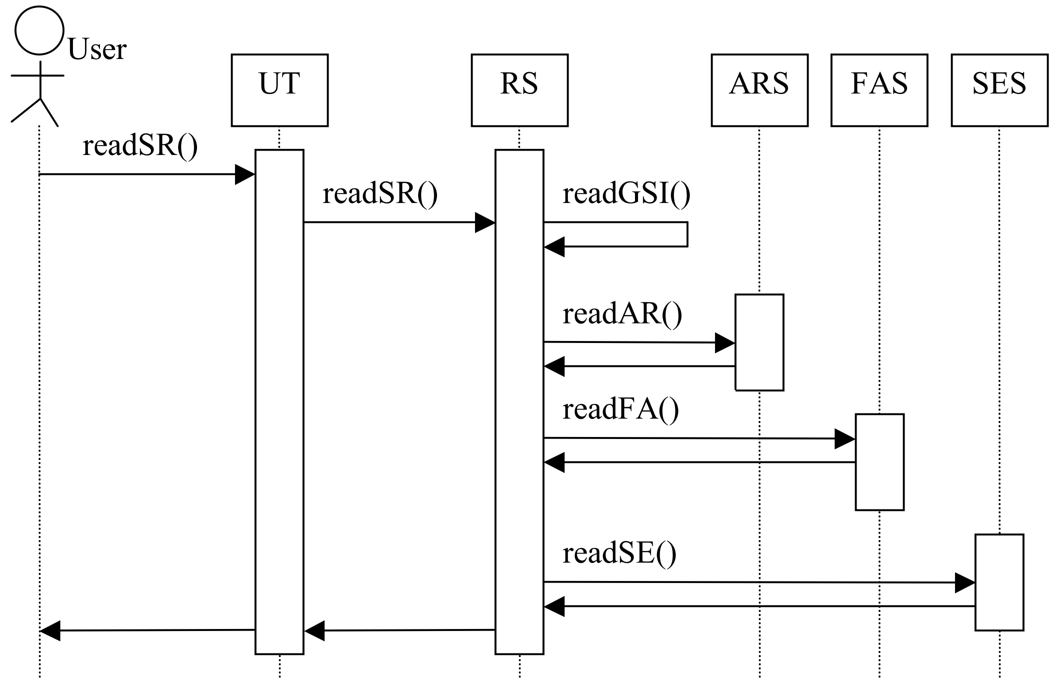

Figure 3-2: Interaction Diagram for Reading Student Records

Save Student Record

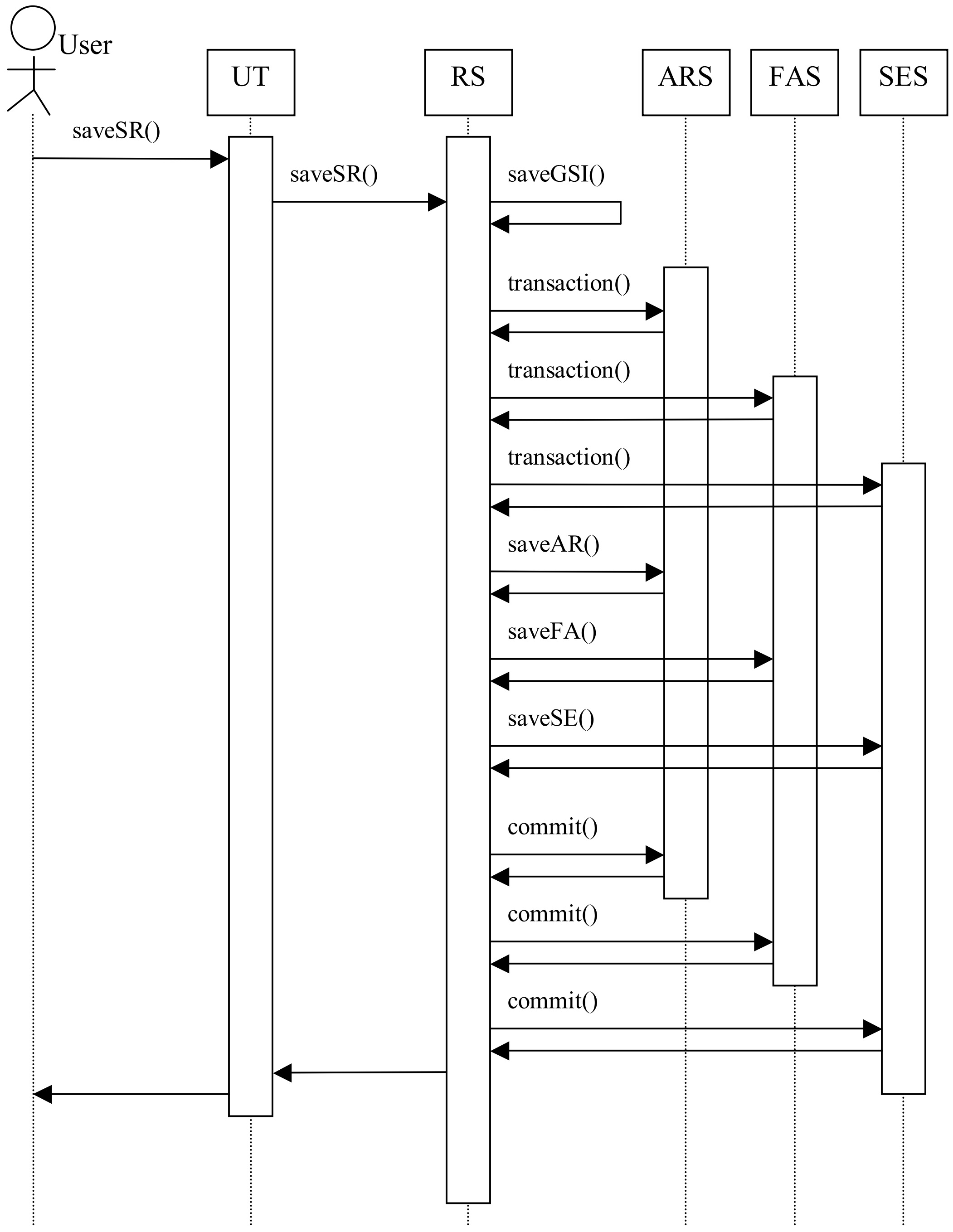

Figure 3-3: Interaction Diagram for Saving Student Records

Add Student

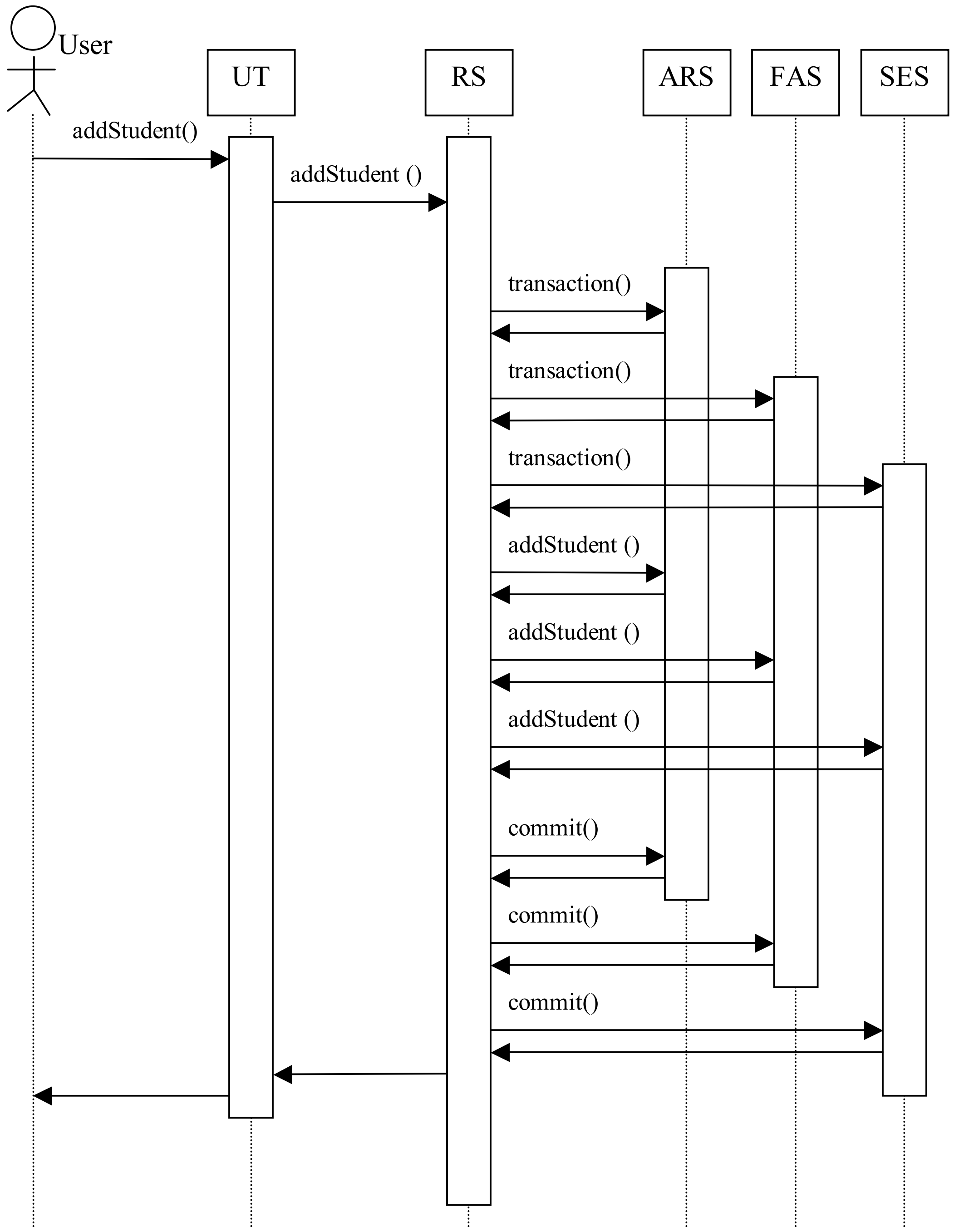

Figure 3-4: Interaction Diagram for Adding Student to the System

The interaction diagrams in Figure 3-2, Figure 3-3 and Figure 3-4 outline the process of carrying out each of the system level commands and how the components interact in this case study. This simplified student information system model will serve as case study throughout this thesis and each of the following chapters will augment portions of this model to illustrate how the UniFrame access control model is constructed and verified. As addressed in Section 3.1, this case study should be sufficient for demonstrating the variety of access control related issues that may appear in systems.

This chapter presents a student information system case study that will be used as the context for discussing the proposed UniFrame access control model developed in this thesis. The following chapter will start the discussion on the details of this access control model and draws upon ideas from model driven access control and the Common Criteria as presented in Chapter 2.

4. COMPONENT SPECIFICATION AND SEARCHING

This chapter proposes a method for extending the URDS to account for access control characteristics. Section 4.1 address the types of information needed to characterize access control properties of components in order to facilitate searching. Section 4.2 explains the algorithm for determining if a component meets the requirements and should be returned in response to the query.

4.1 Access Control Characteristics Search Criteria

As mentioned previously, Model Driven Access Control and the Common Criteria both suggest that key parts of the access control models for components include the naming of resources in the system and the identification of resources that are protected by access control guards in components. It will be shown that these two pieces of information, contained resources and protected resources, can form the basis of a searching mechanism for locating software components with the intended access control properties.

4.1.1 Resource Naming

In order to identify resources that are contained in and protected by components, it will be necessary to develop a method for naming resources. Although naming the resources in a domain will be a key part of the UniFrame knowledge-base for the domain, it is not the goal of this thesis to develop a general naming scheme for application to all domains. Instead, it is assumed that domain experts will come to an agreement for resource naming in each domain. Examples of naming schemes can be found in documents of the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C) concerning Uniform Resource Names (URN) [W3C01, IET01a, IET01b, IET01c, IET99]. From a different perspective, the Resource Access Decision Facility Specification (RADS), from the Object Management Group, addresses resource naming through a standard data structure consisting of a string representing the naming authority and a list of name-value pairs that represent the resource [OMG01b]. However resource names are determined, it is important that each domain has an agreed upon standard as part of the UniFrame domain model.

As an example, consider the university student information systems presented in Chapter 3. The name space for resources in the student information system case study will use the RADS structure for resource naming. The student information system (sis) name space contains a resource called *student record* (*sr*) which then contains four sections: *general_student_information* (*gsi*), *academic_ record* (*ar*), *financial_aid* (*fa*), and *student_employment* (*se*). Each of these four can be further broken down into more fine grained parts as illustrated in Figure 4-1. For instance, the resource name for a particular student's transcript is:

```
ResourceName = {        student_record = fsmith,
                        section = academic_record,
                        part = transcript           }
```

This refers to the transcript for a student with the student Id of student Smith.

Figure 4-1: Student Information System Resource Knowledge-Base

Figure 4-1 represents the structure of resources in the student information system case study. Using this structure for naming allows for specific resource names to be expressed as a single string: *fsmith.academic_record.transcript,* or more succinctly as f*smith.ar.transcript* for the previous example. All academic records can represented as a group using the * wildcard. For instance. *. *ar* represents the academic record for each student, with such an expression being useful in the access control policy to indicate that an academic advisor should have read access to the academic records of all student.

Along with a name, each resource must also be mapped to a set of operations available to perform on the resource. Figure 4-1 makes note of these operations for each resource. With an agreed upon naming convention for resources defined in the UniFrame knowledge-base for the domain, component developers will have a common vocabulary through which to identify resources within the components they create. This resource naming vocabulary forms the basis for creating system access control specifications and querying for concrete components in the URDS.

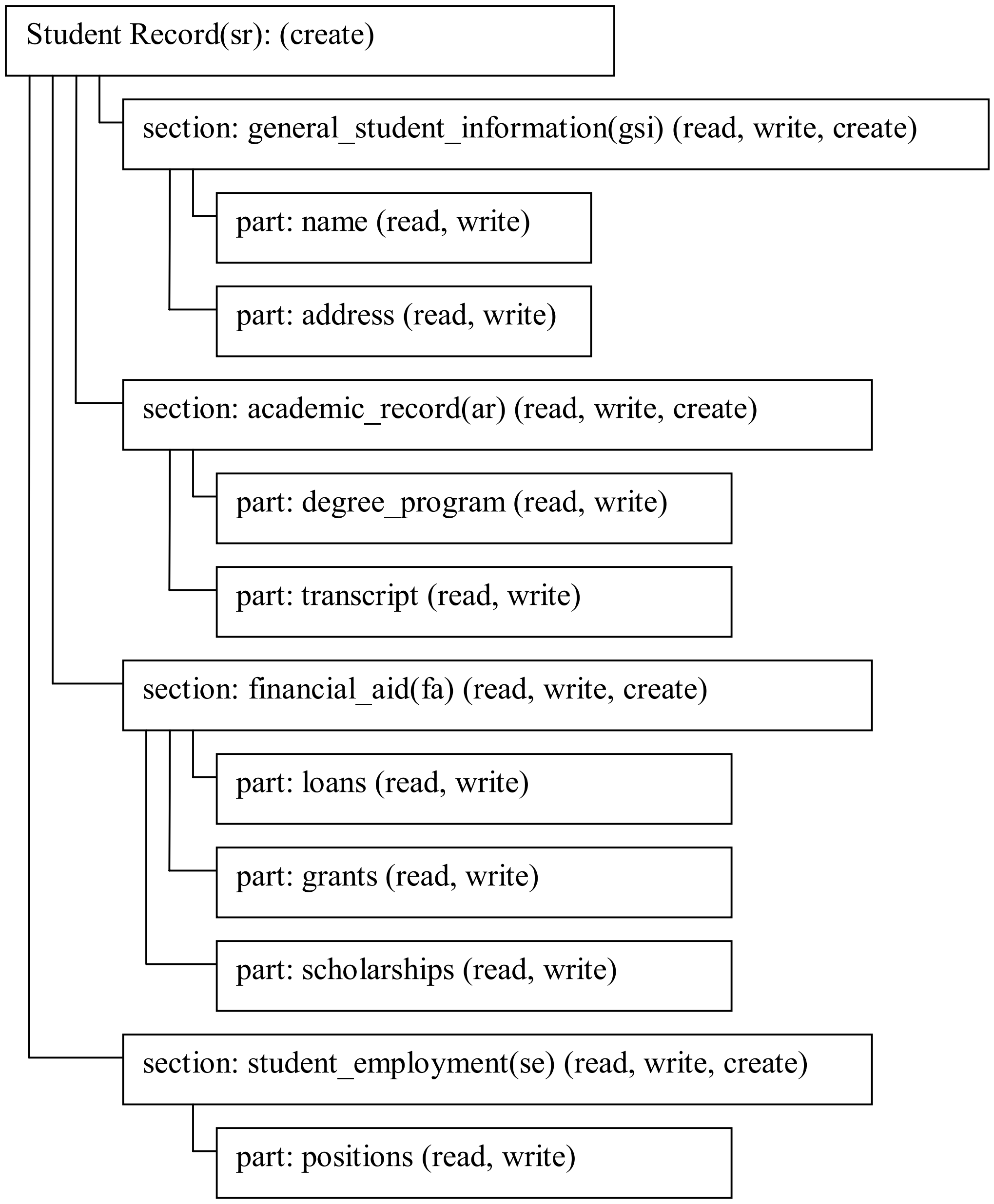
Student Record(sr): (create)
section: general_student_information(gsi) (read, write, create)
part: name (read, write)
part: address (read, write)
section: academic_record(ar) (read, write, create)
part: degree_program (read, write)
part: transcript (read, write)
section: financial_aid(fa) (read, write, create)
part: loans (read, write)
part: grants (read, write)
part: scholarships (read, write)
section: student_employment(se) (read, write, create)
part: positions (read, write)

Figure 4-1: Student Information System Resource Knowledge-Base

4.1.2 Contained Resources

With the knowledge-base defining the resources and operations within a domain, component developers are responsible for informing the system integrator as to which resources are contained within a particular component. In the context of this thesis, we define a *contained resource* to mean that the component carries out operation(s) on the resource within the component. By having each component specifying which resources it contains, it is possible to analyze the scope of a component's affects on resources within the system. It is assumed that resources contained in a component can only be accessed via that component and that another component needing access to those resources will act on those resources via the containing component.

4.1.3 Protected Resource

In addition to specifying the resources contained in a component, component developers will specify which resources are protected by that component via access control guards. In terms of the model driven access control, this information expresses how the component's platform specific model is parameterized with access control guards. Clearly, the *protected resources* in a component will be a subset of the *contained resources* in a component.

4.2 Matching Algorithm

The combination of *contained* and *protected resources* for each component, forms that basis of the matching algorithm for access control properties. The system integrator defines which resource must be protected in the system being created. Therefore, the search criteria will be the set of resources to be protected, *P*. The matching logic proceeds as follows. For each resource requiring protection, determine if the resource is contained within the component under consideration. If the component contains that resource, then determine if the component protects the resource from unauthorized operations. If the preceding examination results in a 'yes' for each resource requiring protection, then there will be a match between that component and the system integrator's requirements. Otherwise the component is discarded.

In the URDS, component matching is carried out by the Headhunters and is based on the type of the component along with several Quality of Service criteria (i.e. throughput, end-to-end delay, etc.). The access control matching will be implemented through logic programming as a series of logical tests to determine if a component meets the required resource protection properties. The following sections provide an introduction to logic programming and develop the matching algorithm used for identifying appropriate components.

4.2.1 Logic Programming

Logic programming languages are declarative languages and are considerably different from the imperative languages that are most commonly used in the business world. Programming in logic programming languages is a nonprocedural and programs do not state exactly how a result is to be computed. Instead they describe the form of the result that is desired. It is assumed that the logic programming inference mechanism can determine how the result is to be achieved. Therefore, logical programming languages supply the environment with both the relevant information and with a means of inference for computing the results. Predicate calculus is the means of supplying information to the computer and resolution is the means of inference [SEB02].

Logic programming is based on the foundation of predicate calculus. Propositions are logic statements stating the relationships between different objects. The simplest form of a proposition is an atomic proposition, and it consists of compound terms. A compound term is one element of a mathematical function notation and consists of two parts: a functor and an ordered list of parameters. The functor is the symbol that names the relation. For instance, consider the following proposition [SEB02].

```
student(amcrespi)
```

It states that {amcrespi} is a 1-tuple in the relation called student which might mean that alex is a student. Propositions have two basic forms: those that are defined to be true and those where the truth has yet to be determined. These are known as facts and queries respectively. In addition, compound propositions are made of two or more propositions joined by logical operators such as conjunction, disjunction, implication, and equivalence.

Beyond propositions supplied by a logic program, inferring new proposition from existing propositions can be done using the process of resolution. For propositions used for resolution, only a restricted type clausal form, called Horn clauses, can be used. Horn clauses can have either a single atomic proposition on the left side or an empty left side. The left side of the Horn clause is referred to as the head and headed horn clauses state relationships while headless horn clauses are used to represent facts.

Relationship: $father(C) \subset parent(C, A) \cap male(C)$
Facts: $parent(henry, alex)$

The preceding relationship reads like a logical implication written backwards. It reads: If C is the parent of A and C is male, then C is a father. The fact preceding fact is intended to indicate that henry is the parent of alex. Therefore a logic program consists of a database of facts and relationships that when queried will return the truth value of the query as derived from the database.

Prolog is one such example of a logic programming language, and Prolog statements are constructed from terms that can be constants, variables, or structures. A constant is either an atom or an integer with atoms being the symbolic values of Prolog. Atoms are either strings of letters, digits, and underscores that begins with a lowercase letter or a string of any printable ASCII characters delimited by apostrophes. A variable name is any string of letters, digits, and underscores that begins with an uppercase letter. Variables are not bound to types by declaration, but instead binding of a value, called instantiation, occurs in the resolution process. Structures are the atomic propositions of predicate calculus and are of the general form: *functor(parameter list).* The parameter list can be any list of atoms, variables, or other structures.

These Prolog statements can be headed or headless Horn clauses and are terminated with a period as shown below [SEB02]. Headless Horn clauses represent facts in the Prolog database whereas headed Horn clauses are used to define relationships.

Headed Horn clauses are written like reverse implication where the expression to the right of :- implies that the expression to the left :- is true.

Relationship: $father(C) :- parent(C, A) \cap male(C).$
Facts: $parent(henry, alex).$

A collection of facts and rules provides the basis of a theorem proving system. A theorem is stated in the form of a proposition that is to be proved or disproved. Such propositions are called goals or queries in Prolog and are syntactically identical to headless Horn clauses. For instance, if the following query is submitted to the inference engine, Prolog would return ‘yes’ or ‘no’, with ‘yes’ meaning that the system has proved the statement to be true in relation to the set of rules and facts provided to the system.

father(henry).

An answer of ‘no’ means that the query was either proved false or that the system was unable to prove or disprove the statement. When a query contains one or more variables, the system will attempt to use unification to find an instantiation of the variables that yields are true value for the query. Prolog will then identify the instantiations of variables that result in the query being true.

In general, there are two processes for matching a query to facts and rules. In one case called forward chaining, the system begins with the facts and rules and attempts through resolution to derive the query from these statements. An alternative system, backward chaining, can start with the query and use resolution to find a sequence of matching propositions that lead back to a known fact in the database. Prolog uses backward chaining.

Prolog supports a basic data structure called a list, which is a sequence of any number of elements with each element being an atom, atomic proposition, or any other term. Lists are enclosed by square brackets, and elements are separated by commas with [] denoting an empty list. As an example, [‘amcrespi’, ‘jdoe’, ‘fsmith’] may denote a list of students [SEB02].

There are several list-based predicates that will be used in this thesis. The append predicate is used to append a list to a list. The syntax is as follows:

```
append(L1,L2,L3).
```

The predicate is true when the L3 is the concatenation of L1 and L2. In addition to append, Prolog has a special notation | for splitting the head of a list from the tail of list. In general [X|Y] denotes a list with head X and tail Y. For instance, [a,b,c] unifies with [Head|Tail] resulting in Head=a and Tail=[b,c]. Prolog also has the facility to determine if something is a member of a list. The predicate *member(X,L)* evaluates to true when X is a member of the list L.

Much like the judicial system where suspects are innocent until proven guilty, Prolog only determines that a query is true if it can be proven from the database or facts and rules. Call the closed-world assumption, if a query cannot be proven true, it is assumed to be false.

4.2.2 Access Control Matching in Prolog

Using Prolog as the mechanism for component searching requires expressing the algorithm in fact and relationships. The system requirements for resources to be protected can be expressed as a list of resource names. For instance, the following fact would say that the general student information and student employment portions of any student record require access control protection. Remember that *'s in this expression indicate any student's record.

```
require_protect(['*.gsi','*.se'']).
```

In addition to specifying what needs protection, there will be a need for facts indicating the contained and protected resources in each component. The following two facts expressed that the concrete component *StudentEmploymentServer1* contains the student employment portion of student records and that the component also ensures that this resource is protected with a guard.

```
contain('StudentEmploymentServer1','*.se').
ensure_protect('StudentEmploymentServer1','*.se').
```

There is a special case that needs to be handled in the case that a component neither contains nor protects any resources. If only the previous facts are used, then components without resources will not be represented in the database by any facts. Therefore, the following predicate is used to express that a component is to be considered for matching.

```
consider('StudentEmploymentServer1').
```

In this way even components that do not contain resources can still be successfully evaluated by querying the Prolog database.

Once the necessary facts are expressed in Prolog, the next step is to develop predicates to support matching of components to the system specifications. The first step in this process is to develop a predicate to determine if a component protects the resources that it contains and if those resources are present in the list of resources requiring protection. The logic is represented as a recursive predicate that successively investigates each resource in the list of resources that requires protection in the system. The base case of the recursion assumes that all components protect an empty list of resources. The predicate returns true if, for each resource requiring protection, the component contains and protects the resource or does not contain the resource.

```
protects(_,[]).
protects(Component,[Resource|OtherResources]) :-
        (contain(Component,Resource),ensure_protect(Component,Resource) ;
          (not contain(Component,Resource))),
        protects(Component,OtherResources).
```

Finally, the following predicate evaluates to true if a component is to be considered and it protects the required resources. It should be noticed that it also returns true for any components that do not contain resources that require access control protection. This is important because components that don’t contain resources that are required to be protected will satisfy the system requirements.

```
match(Component) :- consider(Component),
                    require_protect(Resources),
                    protects(Component,Resources).
```

Together, these facts and relationships provide the mechanism for implementing the access control matching algorithm presented previously. Table 4-1 summarizes this logic program and would successfully match the *StudentEmploymentServer1* component while failing to match *StudentEmploymentServer2* by evaluating the following two queries.

```
match('StudentEmploymentServer1').
match('StudentEmploymentServer2').
```

Table 4-1: Example Access Control Matching Prolog Program

```
require_protect(['*.se']).
consider('StudentEmploymentServer1').
consider('StudentEmploymentServer2').
contain('StudentEmploymentServer1','*.se').
contain('StudentEmploymentServer2','*.se').
ensure_protect('StudentEmploymentServer1','*.se').
protects(_,[]).
protects(Component,[Resource|OtherResources]) :-
   (contain(Component,Resource),ensure_protect(Component,Resource) ;
    not contain(Component,Resource)),
    protects(Component,OtherResources).
match(Component) :- consider(Component),
              require_protect(Resources),
              protects(Component,Resources).
```

4.3 Component UMM Specification

Because the UniFrame Approach uses UMM specifications to describe concrete components, the contained and required resource information must be added to the specification to facilitate component searching. This simply requires expressing the resources that are contained in the component along with resources that are protected. Table 4-2 provides an example UMM specification in which the security auxiliary attribute now contains an access control section which specifies the names of contained and protected resources for the component. The form of this specification indicates that the component contains and protects the general student information portion of any student record. The specifications provide the information regarding contained and protected resources the URDS Headhunters need in order to perform component matching based on access control properties.

Table 4-2: UMM Specifications for Example Concrete Component

Concrete Component: *RecordServer*

- 1. Component Name: *RecordServer1*
- 2. Component Subcase: *RecordServerCase1*
- 3. Domain Name: University
- 4. System Name: Student Information
- 5. Informal Description: Provide a basic student information system for a university.
- 6. Computational Attributes:
    - 6.1 Inherent Attributes:
        - 6.1.1 id: N/A
        - 6.1.2 Version: version 1.0
        - 6.1.3 Author: N/A
        - 6.1.4 Date: N/A
        - 6.1.5 Validity: N/A
        - 6.1.6 Atomicity: Yes
        - 6.1.7 Registration: N/A
        - 6.1.8 Model: N/A
    - 6.2 Functional Attributes:
        - 6.2.1 Function description: store general student information and coordinate requests to additional servers.
        - 6.2.2 Algorithm: N/A
        - 6.2.3 Complexity: N/A
        - 6.2.4 Syntactic Contract
            - 6.2.4.1 Provided Interface: *1StudentManagement*, *IStudentRecord*
            - 6.2.4.2 Required Interface: *IStudentManagement, IStudentRecord, IAcademicRecord, IStudentEmployment, IFinancialAid, ITransaction*
        - 6.2.5 Technology: N/A
        - 6.2.6 Expected Resources: N/A
        - 6.2.7 Design Patterns: NONE
        - 6.2.8 Known Usage: NONE
        - 6.2.9 Alias: NONE
- 7. Cooperation Attributes
    - 7.1 Preprocessing Collaborators:*UserTerminalCase1*
    - 7.2 Postprocessing Collaborators: *AcademicServerCase1, FinancialAidServerCase1, StudentEmploymentServerCase1*
- 8. Auxiliary Attributes:
    - 8.1 Mobility: No
    - 8.2 Security
        - 8.2.1 Access Control
            - 8.2.1.1 Contained Resources:**.gsi*
            - 8.2.1.2 Protected Resources: **.gsi*
    - 8.3 Fault tolerance: *L0*
- 9. Quality of Service
    - 9.1 QoS Metrics: *throughput*, *end-to-end delay*
    - 9.2 QoS Level: N/A
    - 9.3 Cost: N/A
    - 9.4 Quality Level: N/A

This chapter presents the proposed extension to component specifications for allowing access control characteristics to play a role in system specification and component searching in the URDS. In addition, a algorithm for matching requirements to concrete component specifications is presented through the use of logic programming. The next chapter will develop a means for static testing of system functionality with respect to access control properties of components and access control policies imposed on the system. The student information system case study will again provide the context for this discussion.

5. PREDICTING COMPOSED SYSTEM ACCESS CONTROL PROPERTIES

This chapter explores how logic programming is used to model the access control characteristics of individual components as well as systems composed from components. Section 5.1 presents how the access control functional characteristics of a component are expressed in PROLOG. Section 5.2 discusses how these individual component specifications are composed together into a specification of the integrated system. Section 5.3 demonstrates how such a model can be queried to examine the behavior of the system relative to various access control policies.

5.1 Modeling System Access Control Behavior

Once components have been searched for via the URDS and matches are returned, the system integrator is tasked with selecting components for composition into the final system for testing. Before such a composition takes place, the system integrator can perform static tests on the system based on a model of the system created from the component specifications. For instance, [SUN03] presents rules for statically predicting the throughput and end-to-end delay system functionality through composition rules and component throughput and end-to-end delay properties. To extend the static testing approach to access-control-based properties, it will be necessary to develop a means to express the link between component functionality and access control authorizations in such a way that the system level functionality can be investigated with respect to access control constraints. Through such a process, it becomes possible to investigate how the system functionality depends on access control policies before the actual system is composed and deployed.

5.2 PROLOG Representation of Access Control Behavior

The access control functionality of a component can be expressed as logical predicates with one predicate per function in the component's interfaces. The aim is not to simulate the full functionality of the component in Prolog but to instead model the access control related behavior of each function. Such an expression, in combination with a model for the proposed access control policy, will allow the system integrator to determine to what degree the policy limits user execution of component functions. Also because system level actions will involve multiple method calls on multiple components, the predicates can be linked together in chains to model composition of the system. This will also involve modeling user credentials, access control guards, and dynamic attributes as well as access control policy via Prolog. The following sections examine each of these model elements in detail.

5.2.1 Modeling Component Interfaces

Components interact with each other through the methods in their interfaces, and these interactions form the basis of the composition of the system from concrete components. Such a composition can be modeled via logic programming by expressing each method as a predicate. When completing a call to a method on one component involves a call to one or more methods on other components, then evaluating the predicate of the originating method depends on the evaluation of resulting calls on other components. For instance, the following predicate models the *writeStudentRecord* method on a concrete component called *record_server_1*. For this predicate to evaluate to true, each call to the financial aid, academic record, and student employment servers must also succeed.

```
saveStudentRecord(P,S,SR):-
        saveFinancialAid(P,S,SR),
        saveAcademicRecord(P,S,SR),
        saveStudentEmployment(P,S,SR).
```

The *saveStudentRecord* method on the record server in part results in saving the general student information section to the record server. If this saving involves an access control authorization, then the preceding predicate would be modified as follows to include a guard for protecting the write access to the general student information section of the student record.

```
saveStudentRecord(P,S,SR):-
            guard(P, [[sr,S],[section,gsi]],write),
            saveFinancialAid(P,S,SR),
            saveAcademicRecord(P,S,SR),
            saveStudentEmployment(P,S,SR).
```

Using this modeling approach, each component's access control model, described in Prolog, consists of a predicate for each method of that component plus predicates for representing access control guards within that component. If conventions on the naming of predicates are followed, the predicates for all components in a system can be combined into a single Prolog database representing the entire system. For instance, the *saveStudentRecord* predicate for the *UserTerminal* model would include a reference to the *saveStudentRecord* predicate in the *RecordServer* model but both component model cannot use identical names for their predicates or else the logic program will fail to function properly. The following sections discuss other issues related to the Prolog model and then Section 5.2 discusses conventions used in the composition of component models into a system level model.

5.2.2 Modeling Users and User Credentials

Beyond modeling component interfaces, the access control model also includes information about potential users of the system. Each user of the system has credentials that identify security attributes which will be used by the access control model to determine the user's access privileges. This information is modeled using a combination of functors to create a fact that relates a user ID to information such as roles, groups, and access ID. For example,

```
authenticate(fsmith,credential(accessID(fsmith),role([student]),group([student]))).
```

indicates that the user *fsmith* has credentials such that the access Id is *fsmith*, role is *student*, and he belongs to the *student* group. The information in this fact is accessed by the model in two situations. First, to identify a principal in the system which is defined by the following predicate, this predicate is used

```
principal(AID,P):- authenticate(AID,P).
```

to retrieve a principal's credentials from an access ID through queries like principal(fsmith,P) which will bind fsmith's credentials to the variable P during resolution. The model can also identify if a user is a student by examining a user's credentials to see if the student role is present in the list of roles.

```
student(S):- authenticate(S,P),
             arg(2,P,ROLEPRED),
             arg(1,ROLEPRED,ROLE),
             member(student,ROLE).
```

This expression requires some explanation by way of an example to be clear. If the variable S is bound to *fsmith*, then the authenticate functor will bind the variable P to that user's credential's expression.

```
S = fsmith
P = credential(accessID(fsmith),role([student]),group([student]))
```

The *arg* functor is used to pick apart the portions of the credentials to isolate *fsmith*'s roles. For instance, since *P* is bound to the credentials, *arg(2,P, ROLEPRED)* will bind the variable *ROLEPRED* to the second argument in the functor bound to *P*. In this case *ROLEPRED* will be bound to *role([student])*. The variable *ROLEPRED* was named to indicate it was bound to the role predicate. The *arg* functor is again used to isolate the list of roles by binding *ROLE* to the first argument in the functor bound to *ROLEPRED*. This will bind *ROLE* to *[student]*. The final part of the student(s) predicate tests to see if *student* is a member of smith's list of roles in this example. In this way, the predicate returns true if the user contains the role student in his credentials. Together, these facts and relationships act as a model of the authentication portion of the student information system model.

5.2.3 Modeling Access Control Guards

The guard(s) in each component can be modeled in anyway seen fit by the component developer. Depending on the component's model, the guard may consult a dynamic attribute server in order to retrieve any dynamic security attributes that may be needed to make access control decisions. As mentioned in Chapter 2, this particular access control functionality is generally only needed in more advanced systems but is included in this case study to demonstrate how it can be handled. The guard would then consult the access control policy to determine if the access request is authorized. To perform such a check, the guard needs three pieces of information: the principal credentials, the resource name, and the operation requested. These are represented by P, R, and O respectively in this predicate:

```
guard(P,R,O):- dynamicAttributes(P,NEWP,R),
               accessControl(NEWP,R,O).
```

The NEWP variable represents the modified set of attributes returned from the dynamic attribute server and this new set is used when evaluating the access request. This modification is necessary to account for dynamic attributes in the system.

5.2.4 Modeling Data and Dynamic Attributes

In order to model the action of a dynamic attribute server, there may be a need to model some of the data stored in a component. As mentioned previously, at times access control decisions are based on context information such as time, ownership, or geographic location. In the case of the student information system, students may be allowed to access their own records. In this case, there must be a way to determine if a user is the owner of a student record. Each component that stores information may potentially need a way to model this information in Prolog to such a degree that dynamic security attributes are properly modeled. A component developer should supply examples of the data and documentation to allow the testers to adapt the data to their testing needs.

In this case study, the necessary dynamic attribute information is retrieved from the resource name since in this example the student record is linked to the student through the student's Id. If the principal's access Id is the same as the record's student Id, then the dynamic attribute predicate below will append the role of student owner to the security attributes.

```
DynamicAttributes(P,NEWP,R):-
        arg(1,P,AIDPRED), arg(1,AIDPRED,AID),
        arg(2,P,ROLEPRED), arg(1,ROLEPRED,ROLE), arg(3,P,GROUPPRED),
        (member([sr,AID],R) ->  append(ROLE,[studentowner],ROLE2);
                                append(ROLE,[],ROLE2)),
        (member([noteowner,AID],R)->    append(ROLE2,[noteowner],ROLE3);
                                        append(ROLE2,[],ROLE3)),
                                        NEWP=credentials(AIDPRED,role(ROLE3),
                                                         GROUPPRED).
```

How exactly each concrete component models these actions will depend on the component developer.

5.2.5 Modeling Access Control Policy

The UniFrame access control model assumes a uniform access control policy for all components in a system. For the Prolog access control model, where this policy is stored and how is it cached or accessed by components it not considered. The goal is only to determine how particular policies impact the ability of users in a system to perform certain actions. Therefore, evaluating the access control policy in the Prolog model is done through a single predicate called *accessControl(P,R,O)* with P, R and O representing the principal's security attributes, resource name, and operation respectively. The system integrator is free to design this predicate to represent the desired policy. Depending on the implementation of Prolog interpreter used, it may also be possible to have this predicate call an external access control policy evaluator rather than using Prolog predicates to define the policy. This would be useful in that the actual access policy and policy evaluators to be used in the deployed system could be used in the model instead of a handcrafting a Prolog version of the policy and policy evaluator. This will reduce the likelihood of errors in evaluating the system model.

5.3 Model Composition

To facilitate the composition of individual component specifications into a system level specification, a series of conventions are used to avoid conflicts in the naming of predicates when component predicates are combined together. First, each component in the model is assigned a unique name. This assigned name is used to avoid the possibility of two components having the same name. The predicate for each of a component's methods is given a name consisting of the concatenation of the component's name and the method's name. This will eliminate the possibility of having a collision between the definitions of predicates from two different components. The previous *saveStudentRecord* example when created by a component named *record_server_1*, would result in the following predicate.

```
record_server_1_saveStudentRecord(P,S,SR):-
                guard(P, [[sr,S],[section,gsi]],write),
                financial_aid_server_1_saveFinancialAid(P,S,SR),
                academic_record_server_1_saveAcademicRecord(P,S,SR),
                student_employment_server_1_saveStudentEmployment(P,S,SR).
```

Notice that each of the saveFinancialAid, saveAcademicRecord, and saveStudentEmployment predicates is prefixed with their component names. In this way,name collisions can be avoided. Table 5-1 through Table 5-5 provide example component specifications for each of the five concrete components needed for the student information system case study. The terms in << >> are place holders to be replaced with the names assigned to each component in the system. In addition to predicates for each component, the composed system model will include the authentication predicates for principals as described previously along with user credentials and access control policy logic. Table 5-6 and Table 5-7 provide examples of these predicates.

Table 5-1: Example User Terminal Model

```
<<USER_TERMINAL>>readStudentRecord(AID,S):-
        principal(AID,P), student(S), <<RECORD_SERVER>>readStudentRecord(P,S).

<<USER_TERMINAL>>saveStudentRecord(AID,S,SR):-
        principal(AID,P), student(S), <<RECORD_SERVER>>saveStudentRecord(P,S,SR).
```

Table 5-2: Example Record Server Model

```
<<RECORD_SERVER>>readStudentRecord(P,S):-
        <<RECORD_SERVER>>Guard(P,[[sr,S],[section,notes]],read));
        <<FINANCIAL_AID_SERVER>>readFinancialAidRecord(P,S);
        <<ACADEMIC_SERVER>>readAcademicRecord(P,S);
        <<STUDENT_EMPLOYMENT_SERVER>>readStudentEmploymentRecord(P,S);
        (<<RECORD_SERVER>>Guard(P,[[sr,S],[section,gsi]],read)).


<<RECORD_SERVER>>saveStudentRecord(P,S,SR):-
          ((member(gsi,SR),
           <<RECORD_SERVER>>Guard(P,[[sr,S],[section,gsi],[all,all]],save));
           not member(gsi,SR)),
          ((member(ar,SR) ,
           <<FINANCIAL_AID_SERVER>>saveFinancialAidRecord(P,S,SR));
            not member(ar,SR) ),
          ((member(fa,SR) ,
           <<ACADEMIC_SERVER>>saveAcademicRecord(P,S,SR));
           not member(fa,SR) ),
          ((member(sf,SR),
           <<STUDENT_EMPLOYMENT_SERVER>>saveStudentEmploymentRecord(P,S,SR));
           Not member(sf,SR) ).

<<RECORD_SERVER>>Guard(P,R,O):-
          <<RECORD_SERVER>>DynamicAttributes(P,NEWP,R),
          accessControl(NEWP,R,O).

<<RECORD_SERVER>>DynamicAttributes(P,NEWP,R):-
          arg(1,P,AIDPRED),arg(1,AIDPRED,AID),
          arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
          arg(3,P,GROUPPRED),
          (member([sr,AID],R)->
           append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
          (member([noteowner,AID],R)->
           append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
          NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).
```

Table 5-3: Example Academic Record Server Model

```
<<ACADEMIC_RECORD_SERVER>>readAcademicRecord(P,S):-
            <<ACADEMIC_RECORD_SERVER>>Guard(P,[[sr,S],[section,ar]],read).

<<ACADEMIC_RECORD_SERVER>>saveAcademicRecord(P,S,AR):-
            <<ACADEMIC_RECORD_SERVER>>Guard(P,[[sr,S],[section,ar]],save).

<<ACADEMIC_RECORD_SERVER>>Guard(P,R,O):-
            <<ACADEMIC_RECORD_SERVER>>DynamicAttributes(P,NEWP,R),
            accessControl(NEWP,R,O).

<<ACADEMIC_RECORD_SERVER>>DynamicAttributes(P,NEWP,R):-
            arg(1,P,AIDPRED),arg(1,AIDPRED,AID),
            arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
            arg(3,P,GROUPPRED),
           (member([sr,AID],R)->
             append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
           (member([noteowner,AID],R)->
             append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
            NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).
```

Table 5-4: Example Financial Aid Server Model

```
<<FINANCIAL_AID_SERVER>>readFinancialAid(P,S):-
            <<FINANCIAL_AID_SERVER>>Guard(P,[[sr,S],[section,fa]],read).

<<FINANCIAL_AID_SERVER>>saveFinancialAid(P,S,SLR):-
            <<FINANCIAL_AID_SERVER>>Guard(P,[[sr,S],[section,fa]],save).

<<FINANCIAL_AID_SERVER>>Guard(P,R,O):-
            <<FINANCIAL_AID_SERVER>>DynamicAttributes(P,NEWP,R),
            accessControl(NEWP,R,O).


<<FINANCIAL_AID_SERVER>>DynamicAttributes(P,NEWP,R):-
            arg(1,P,AIDPRED),arg(1,AIDPRED,AID),
            arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
            arg(3,P,GROUPPRED),
           (member([sr,AID],R)->
             append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
           (member([noteowner,AID],R)->
             append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
            NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).
```

Table 5-5: Example Student Employment Server Model

```
<<STUDENT_EMPLOYMENT_SERVER>>readStudentEmployment(P,S):-
        <<STUDENT_EMPLOYMENT_SERVER>>Guard(P,[[sr,S],[section,se]],read).


<<STUDENT_EMPLOYMENT_SERVER>>saveStudentEmployment (P,S,SLR):-
        <<STUDENT_EMPLOYMENT_SERVER>>Guard(P,[[sr,S],[section,se]],save).


<<STUDENT_EMPLOYMENT_SERVER>>Guard(P,R,O):-
        <<STUDENT_EMPLOYMENT_SERVER>>DynamicAttributes(P,NEWP,R),
        accessControl(NEWP,R,O).


<<STUDENT_EMPLOYMENT_SERVER>>DynamicAttributes(P,NEWP,R):-
        arg(1,P,AIDPRED),arg(1,AIDPRED,AID),
        arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
        arg(3,P,GROUPPRED),
       (member([sr,AID],R)->
         append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
       (member([noteowner,AID],R)->
         append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
        NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).
```

Table 5-6: Authentication and User Credentials

```
authenticate(acrespi, credential( accessID(acrespi), role([student]), group([student]))).

authenticate(jdoe, credential(accessID(jdoe), role([bursar]), group([staff]))).

principal(AID,P):- authenticate(AID,P).


student(S):- authenticate(S,P), arg(2,P,ROLEPRED), arg(1,ROLEPRED,ROLE), member(student,ROLE).
```

Table 5-7: Example Access Control Policy

```
accessControl(P,R,O):- arg(3,P,GROUPPRED),arg(1,GROUPPRED,GROUP),
                (member(student,GROUP)->
                 checkstudent(ROLES,R,O);checkstaff(ROLES,R,O)
                ).

%*** evalute policy for students trying to access the records ***
checkstudent(ROLES,R,O):- R=[SID,SECTION,PART|_],
                ( ( (member(studentowner,ROLES),
                   (not SECTION==[section,notes]),
                    O==read);

                   (member(studentowner,ROLES),
                   SECTION==[section,gsi],
                   O==save)
                  ) ->true
                ).


%*** evaluate policy for staff try to access the records ***
checkstaff(ROLES,R,O):- R=[SID,SECTION,PART|_],
                 (
                     (    (member(bursar,ROLES),
                     SECTION==[section,gsi],
                     O==read);

                     (member(bursar,ROLES),
                     SECTION==[section,se],
                     O==read);

                     (member(bursar,ROLES),
                     SECTION==[section,se],
                     O==save);

                     (member(bursar,ROLES),
                     SECTION==[section,fa],
                     O==save);

                     (member(academicadvisor,ROLES),
                     SECTION==[section,gsi],
                     O==read);

                     (member(academicadvisor,ROLES),
                     SECTION==[section,ar],
                     O==read);

                     (member(academicadvisor,ROLES),
                     SECTION==[section,ar],
                     O==save)
                     ) ->true
                ).
```

The process of composition is a process of concatenating these preceding predicates into a single document and replacing the << >> labels with the concrete component names. To automate this process, each deployed concrete component will have an additional method allowing the system integrator to retrieve the logic predicate such that they are formatted appropriately for the system being generated. This method is called *getComponentACModel* and requires two parameters: the name assigned to that component, and a list of post-processing collaborators and their assigned names. With this information, the component will construct and return a string containing the Prolog model for its access control behavior. Table 5-8 contains example code that implements this function for a user terminal component.

Table 5-8: getComponentACModel Implementation for a User Terminal Component

```
public String getComponentACModel(String componentName,
                        Hashtable postProcessingCollaborators)
{

  StringBuffer prologBuffer = new StringBuffer();

  prologBuffer.append("%**********"+componentID+"**********\n");
  prologBuffer.append(componentName+
                "_readStudentRecord(AID,S):-principal(AID,P)," +
                "student(S),");

  //read in record server name from post processing collaborators list
  String recordServerName = (String)
  postProcessingCollaborators.get("recordserver");

  prologBuffer.append(recordServerName);
  prologBuffer.append("_readStudentRecord(P,S).\n");
  prologBuffer.append(componentName+
                "_saveStudentRecord(AID,S,SR):- "+
                "principal(AID,P),student(S),");
  prologBuffer.append(recordServerName);
  prologBuffer.append("_saveStudentRecord(P,S,SR).\n");
}
```

5.4 Querying the Access Control Model

With a properly composed logic program representing the student information system, it becomes possible to explore the types of system functionality allowed to users in the system based on component access control characteristics, access control policy, and user security attributes. Such an exploration is accomplished via querying the Prolog database created to model the system's access control behavior. The user terminal acts as the point of contact for a user with the system and provides two predicates for querying the system:

```
record_server_1_readStudentRecord(P,S).
record_server_1_saveStudentRecord(P,S,SR).
```

In both predicates, P represents the access Id of the user accessing the system and S represents the Id of the student whose record is being accessed. The SR variable in the second predicate represents the portions of the student record being written to. Therefore by setting SR to the list *[gsi,se]*, the model would check to see if the user would be allowed to save the general student information and student employment sections of a student record.

For example, Table 5-6 lists the credentials for a user 'jdoe' indicating that this user is part of the university staff and fills the role of bursar. To determine if 'jdoe' can successfully read the student record for 'acrespi' from the system, the following query is submitted to the database.

```
user_terminal_1_readStudentRecord(jdoe,acrespi).
```

Prolog would return a yes or no response to this query based on the resolution process. Instead of a very specific query regarding a particular student's record, it is possible to determine which student records an individual can access by using a variable in the query to represent the student. Given this query, prolog would respond with all bindings to the variable Student that result in yes evaluations of the query.

```
user_terminal_1_readStudentRecord(jdoe,Student).
```

Similar queries can be performed with respect to the save student records functionality. For instance, the following query determines if 'jdoe' is allowed to save the general information portion of the student record for 'acrespi'.

```
user_terminal_1_saveStudentRecord(jdoe,acrespi,[gsi]).
```

The following query tests to see if 'jdoe' can save both the general student information and student employment information to the student record for 'acrespi'. The sections of the student record being written to are represented by a list *[gsi,se]*.

```
user_terminal_1_saveStudentRecord(jdoe,acrespi,[gsi,se]).
```

The response to these queries will depend on the access control guards positioned in the components, component interactions, and also the access control policy provided to the system. Using this type of model, the system integrator or tester can evaluate the ability of a proposed system to control the access of users to system resources and system level functionality.

This type of testing is important for two different aspects of the system's lifecycle. First, querying allows a system tester to evaluate the ability of a system to regulate accesses before the actual system is created. This can speed the process of evaluating combinations of components selected to composition. Second, for a given existing system, the querying allows for changes in access control policy to be evaluated before a new policy is deployed in the actual system. This can increase the likelihood that a modified access control policy will act in the desired fashion when deployed. Both of these validation steps provide a means of static testing of the access control behavior of systems.

The chapter develops a method for using logic programming to perform static system testing on access control properties before systems are actually composed. This static testing allows the system integrator or tester to explore system level functionality with respect to component access control properties and access control properties. Although this provides additional information with regards to component selection, it does not explore dynamic system properties that may be affected by changes in access control properties. Chapter 6 explores the creation of an additional system model based on TLA+ which allows for determining if the access control system can induce deadlocking and evaluate in more detail the dynamics of the system.

6. VERIFICATION OF ACCESS CONTROL BEHAVIOR

This chapter explores how the Temporal Logic of Actions can be used to model component-based distributed systems including the access control behavior in order to determine the correctness of the system composition. TLA+ can examine various safety and liveness properties of a system model. The organization of this chapter is similar to Chapter 5 starting with Section 6.1, which expands on the brief overview of TLA+ from Chapter 2. Section 6.2 presents how the functional characteristics of a component are expressed in TLA+. Section 6.3 discusses how the specifications of these individual components are knitted together into system specifications. Then Section 6.4 demonstrates how the TLA model checker can be used to examine the system specifications by testing for deadlock and verifying liveness properties. Whereas the goal of the previous logic programming model of the system was to help validate how access to system resources depended on access control guards and policies, the TLA+ model developed in this chapter aims to help validate more dynamic system properties to make sure that access control behaviors and policies do not detrimentally affect the dynamics of component interactions.

6.1 TLA Modeling

As mentioned in Chapter 2 [ABA93], the specification of a system can be reduced mathematically to one TLA+ formula:

$$\mathrm{Spec} \overset{\Delta}{=} \mathrm{Init} \wedge \Box[\mathrm{Next}]_{\_\mathrm{variables}} \wedge \mathrm{Liveness}$$

Init represents an initial predicate that specifies the possible initial values of all variables in the system. *Next* is a predicate that expresses the relation between the current state of all variables and the next state(s) of the system. The temporal operator □ (pronounced box) asserts that a formula is always true. Therefore $\square[Next]_{variables}$ states that *Next* is true for every step in the system's behavior. Given this, $Init \wedge \square[Next]_{variables}$ is true of a behavior iff the initial state satisfies *Init* and every step satisfies *Next. Liveness* represents a predicate expressing the liveness requirements. Without liveness properties, there is no guarantee that any action will ever be taken by the system. As an example, the weak fairness expression $WF_{variables}(A)$ says that "it is always the case that, if A is enabled forever, than an A step eventually occurs." [LAM03] To make sure that the system actually performs actions, the predicate for *liveness* can be written as the following.

```
Liveness == WF_variable(Next)
```

This liveness property will make sure that the system being modeled in fact will execute enabled actions. Without this property, the system specification will not require the model to do anything at all.

Modeling the component and the system behavior, from the access control perspective, will require a variety of TLA notations and expressions, and an in depth presentation TLA can be found in [LAM03]. For the purpose of this thesis, the focus will be on functions, records, conditional statements, sets, and state changes with each is explained in the following sections.

6.1.1 TLA Specification Overview

TLA specifications are written in modules with each module requiring its own file. The basic format of a TLA specification file follows the format shown in Table 6-1 through an example that models a set of users attempting to change the value of a variable through addition. For the TLA model checker to function properly, a configuration file must be created that specifies the value of each constant in the specification and identifies the predicate representing the specification. The configuration file may also identify the predicate(s) representing invariants in the specifications that should be checked.

The overall format of the specification file simple. The EXTENDS reserved word instructs the model checker to load one or more additional modules that contain predicates to be used in the current module. In this case, the Naturals model is used to define addition. Next the CONSTANTS reserved word specifies unchanging values used in the specification. Values for constants are defined in the configuration file for the module as shown in Table 6-2. The state of a model is represented by the various values of the model's variables and the VARIABLES reserved word is used to define the variables used in the model. It is important to know that variables defined in the modules extending this module are also part of the model's state. The rest of the specification consists of predicates expressed in TLA notation. The modeler uses the configuration file to define which predicate represented the specification to be checked and perhaps which predicate(s) represent invariants in the model.

Table 6-1: Example TLA Specification (example.tla)

```
------------------------------------ MODULE example ---------------------------------------
EXTENDS Naturals
CONSTANTS ValueSet, UserNames, AllowedUsers
VARIABLES  value, username

TypeInvariant == /\ value \in ValueSet            \* restriction on value
                 /\ username \in UserNames        \* restriction on username

Init == /\ value =0                      \* initialize value to zero
        /\ username \in  UserNames       \* initialize usename to a member of set UserNames

Check(u) == u \in  AllowedUsers      \*operator and policy for determining access

Change == /\ Check(username)         \* if allowed, add one to value
          /\ value' = value + 1
          /\ UNCHANGED<<username>>

Next  ==  /\ Change
Spec  ==  Init /\ [][Next]_<< value, username>>
----------------------------------------------------------------------------------------------------
```

Stepping through each predicate shown in Table 6-1 in order, the *TypeInvariant* expression states that at all times the variables *value* and *username* must be elements of the sets *ValueSet* and *UserNames*, respectively. For the model checker to validate the model, this must be true in all of the states that are reachable. The next predicate *Init* is used to define the initial state(s) of the model. In this case, there will be three initial states, one for each user is the *UserNames* set. The $\wedge$ operator represents the logical AND of the components of this predicate. The *check(u)* operator is used as a rudimentary access control point. If the *u* is in the set of allowed users, this operator returns true.

The *Change* predicate is a conjunction of three expressions. The first expression models the access control check by seeing if the user is in the *AllowedUsers* set. The second expression changes the state of the variable *value.value'* represents the variable's value in the next state and *value* represents the value in the current state. This particular expression is true when the new value is one higher than the current value. The third expression dictates that the variable *username* remains unchanged. The notation << >> is used to represent the tuples with any number of elements separated by commas. The *Change* predicate is only enabled when these expressions are true. The *Next* predicate is redundant in this case and is usually used to represent the conjunction or disjunction of several predicates in the specification. Since in this case there is only one action in the model, the *Next* predicate simply equates to the *Change* predicate. It is included here because in more complex specifications, the *Next* predicate will be used in all cases.

In this example, the entire model specification is represented by the predicate *Spec* which is the conjunction of the *Init* predicate and a temporal expression $\Box$*[Next]* which indicates that the predicate *Next* is to be enabled at all times. If *Next* were not always true, the system would be in deadlock. A careful reader will determine that the example specification will result in a deadlock when the initial state with username equal to *ggeorge*. Because *ggeorge* is not included in the *AllowedUsers* set, *check("ggeorge")* will evaluate to false leading to *Next* being false resulting in a deadlock because no action can be taken. The reader might also notice that each initial state will eventually result in the invariant being violated because *value* will increase beyond the values in *ValueSet*. Therefore the model checker will reject this model on both accounts.

Table 6-2: Example TLA Configuration File (example.cfg)

```
SPECIFICATION Spec
INVARIANT TypeInvariant
CONSTANTS
  ValueSet    = {1,2,3,4,5,6,7,8,9,10}
  UserNames   = {"acrespi","hblake","ggeorge"}
  AllowedUsers= {"acrespi","hblake"}
```

Table 6-3 lists the specification such that these two problems are removed by modifying the *Change* predicate and adding a new predicate called *NoChange*. The *Change* predicate now results in *value* cycling through values from 0 to 9 by using the IF-THEN-ELSE construct to control how the value is changed. This makes sure that the invariant is not violated with respect to *value* by keeping its value within the prescribed set. The *NoChange* predicate in effect does not change the state of *value*. *Next* now becomes the disjunction of *Change* and *NoChange* and will always be true and not causing a deadlock.

Table 6-3: Example of Improved TLA Specification (example.tla)

```
------------------------------ MODULE example ---------------------------------
EXTENDS Naturals
CONSTANTS ValueSet, UserNames, AllowedUsers
VARIABLES  value, username

TypeInvariant == /\ value \in ValueSet
                 /\ username \in UserNames

Init == /\ value =0
        /\ username \in  UserNames

Check(u) == u \in  AllowedUsers

Change == /\ Check(username)                        \* is allowed, increment value or change to 0.
          /\ value' = IF value#10 THEN value+1 ELSE 0
          /\ UNCHANGED<<username>>

NoChange == UNCHANGED<<value,username>>  \* when not allowed, value does not change.

Next  ==   \/ Change
           \/ NoChange

Spec  ==  Init /\ [][Next]_<< value, username>>
---------------------------------------------------------------------------------------
```

The preceding specification does not include any liveness properties to ensure that the model in fact executes actions. Adding the conjunction /\ *WF*$_{<<value,name>>}$*(Next)* to the *Spec* expression takes care of this issue by requiring the *Next* action to eventually be taken.

The previous example introduces the general syntax and structure of a TLA specification. Beyond the notation introduced in this example, there are several other aspects of TLA that will be important in developing models for this thesis: records and functions. Records allow a concise notation for grouping information together in a single variable. For instance, one way to model a communications channel between components involves maintaining state information about the channel to determine if data is being transmitted or received. Assume that the channel can have the three states *ready*, *sent*, and *replied* along with data. A channel that is available for transmitting a request can be represented as the following record. The |-> symbol is used to assign a value to each part of a record.

```
[ state|->"ready", data|->""]
```

A component can send data out on a channel by setting the data and changing the state to sent.

```
[ state|->"sent", data|->"read student record"]
```

The component would then wait until the channel state changes to replied, at which time the channel data will contain a reply from the other component.

```
[ state|->"replied", data|->"student record data"]
```

If the variable *channel* contained the previous record, then the notation for accessing the state of the channel is *channel.state*. There is one variation of this notation that makes it easier to change a single record element while leaving the rest unchanged, which helps in writing more concise specifications. In this expression, ! refers to the variable *channel.*

```
channel' = [channel EXCEPT !.state="ready"]
```

This expression will make the value of *channel* in the next state equal to its current value except that the *state* field will be set to *ready*.

There is another notation for specifying records, or in this case a set of records. Given the set Users={“jdoe”, ”acrespi”, ”fsmith”} and the set Commands={“load”, ”save”}, The following expression creates a set containing records with all combinations of elements of the two sets.

```
commands = [u:Users, c:Commands]
```

The following statement represents the same specification but would require editing each time either of the two sets is modified.

```
commands = {   [u|->"jdoe",      c|->"load"]
               [u|->"jdoe",      c|->"save"]
               [u|->"acrespi",   c|->"load"]
               [u|->"acrespi",   c|->"save"]
               [u|->"fsmith",    c|->"load"]
               [u|->"fsmith",    c|->"save"]  }
```

In addition to records, this thesis makes use of the function construct for modeling various forms of data in the system. For example, in the previous chapter, the user credentials were modeled via Prolog predicates and included access Id, roles, and groups. The same information can be modeled in TLA using a function. In the following expression, the variable *attribute* is set to a function to store user security attributes.

```
attribute = ( "accessid" :> {"jdoe"}     @@
              "roles"    :> {"bursar"}@@
              "groups"   :> {"employee"}       )
```

Each member of the domain of the function is mapped to a member of the range of the function via the *:>* symbol. Therefore, mapping the string *“roles”* to a set of roles is accomplished through the expression *“roles” :> {“bursar”}*. In this case, *“roles”* is mapped to a set containing only the string *"bursar"*. The @@ symbol is use to group together multiple relationships into a single function. Accessing the set of roles is accomplished by writing *attribute["roles"]*. This function notation can be used to extend the channel to allow multiple sub-channels in order to allow multiple threads or processes to communicate through the same channel. In essence, the channel becomes a function with as many elements in the domain of the function as there are sub-channels in the channel. Given a set Threads= {1,2,3,4,5} that contains an element for each thread, the following expressions makes *channel* a function that maps elements of the domain of Threads to empty channels that are ready to be used.

```
channel = [t \in Threads |-> [state|->"ready", data |->""] ]
```

In TLA notation, *\in* represents "element of" so the expression would be read as "for every *t* in *Threads* map it to the record *[state|->"ready", data |->""]* ". The resulting value of *channel* could have been written in the following manner, but the previous notation is much more concise.

```
channel = (      1 :> |-> [state|->"ready", data |->""]    @@
                 2 :> |-> [state|->"ready", data |->""]    @@
                 3 :> |-> [state|->"ready", data |->""]    @@
                 4 :> |-> [state|->"ready", data |->""]    @@
                 5 :> |-> [state|->"ready", data |->""]  )
```

6.2 TLA Student Information System Framework

The preceding TLA constructs are used in modeling the student information system components and system developed in the coming sections. The TLA specification for the entire student information system is quite lengthy, making it difficult to clearly demonstrate the framework for assembling components into the complete model. Therefore, this section provides an overview of how the communications between components is modeled, how the access control policy is modeled and evaluated, how the specification models threads, and what the conventions are that are used for naming predicates and variables to avoid naming conflicts between components. In this chapter, a simplified two-component model of the student information system is used as an example because it demonstrates all of the key features while reducing the total size of the model. This simplified model consists of a user terminal and a record server that only contains general student information, and the system will only support reading student record information. This example will be sufficient to demonstrate the process of developing component specifications and composing system specifications. Chapter 7 will use the complete five-component models for validation purposes.

6.2.1 Naming Conventions

In order to avoid collisions between variables and predicates in different components, a naming scheme similar to that used in Chapter 5 is used. Predicates and variables available only to individual components are prefixed with a string to indicate the component they belong to. In the case study, the user terminal predicates are prefixed with *UT_*, and the user terminal variables are prefixed with *ut_*. The channel variables are prefixed with a string for each component that they connect. For instance, the channel between the user terminal and record server that handles read commands is named *ut_rs_read*. Beyond helping to avoid name collisions, this naming convention makes it easier to identify the scope of variables in the system during composition.

6.2.2 Inter-Component Communications

The communications channel between each component is modeled as a record, much like that discussed previously, but with some modification to handle multiple threads. The example components in the case study also communicate via synchronous method calls, and therefore, the channels must support this type of protocol. This requires one channel variable for each communications path between components. In this case study, the user terminal can only read student records on the record server. In the TLA model, this is modeled as one channel, and the following TLA expression initialize the values of these this channel variable.

```
ut_rs_save = [t \in Threads |-> [      state      |->"ready",
                                       data       |->[p|->("":>""),r|->"",c|->"", d|->""],
                                       message |->("":>"")] ]
```

In accordance with the naming conventions, this variable is accessed by both the user terminal and record server so the names are prefixed with "ut_rs". Each channel is actually a function with a record for each element of the set Threads, allowing for multiple concurrent communications channels to be active at any given time. Each channel is represented by a record with state, data, and message fields. The data field is also a record which includes information concerning the principal (p), the student record being accessed (r), the command to execute (c), and any data needed to execute the command (d). The message field is included to allow the replying component to send information back to the caller. Initially, all of the channels are set to the *ready* state and contain no information. A total of eight channel variables will be used to model communications between the five components. The state of the channel used to model the communications protocol is outlined in Table 6-4.

Table 6-4: Communications Channel States

State	Explanation
Ready	The channel is available for sending a request to another component.
Sent	The sending component has just sent a request.
Replied	The responding component has replied to the request.

The following expression checks to see if the state of thread *t* in channel *ut_rs_read* is ready. Remember that *ut_rs_read* is a function with domain Thread and the range contains channel records with one channel per thread.

```
ut_rs_save[t].state = "ready"
```

A component responds to the caller by changing the state to replied and sending message back to the caller in the following manner.

```
ut_rs_save'=[ut_rs_save EXCEPT ![t]=[      state |-> "replied",
                                           data  |-> ut_rs_save[t].data,
                                           message |-> ("response" :> "student information"
                                       ]
              ]
```

Through actions like these, the communications between components can be modeled in TLA.

6.2.3 User Credentials

As with the logic programming model, user credentials must be modeled to provide a means of communicating information needed to make access control decisions. In the proposed TLA model, each user’s security attributes are modeled as the function shown earlier:

```
attribute = ( "accessid"     :> {"jdoe"}    @@
              "roles"        :> {"bursar"}@@
              "groups"       :> {"employee"}      )
```

All of the users of the system are modeled as a set of functions to represent the set of all users. Additional security attributes can easily be added by adding to the domain of the function for each user.

```
attributes = {     ( "accessid"  :> {"hblake"}  @@
                     "roles"     :> {} @@
                     "groups"    :> {"student"}
                   ),
                   ( "accessid":> {"fburns"}  @@
                     "roles"     :> {"bursar"} @@
                     "groups"    :> {"staff"}
                   ),
                   ( "accessid":> {"cwinchester"}  @@
                     "roles"     :>  {"academicadvisor"} @@
                     "groups"    :> {"staff"}                    )    }
```

Accessing the security attributes for a particular user is accomplished via the TLA CHOOSE operator which allows for picking particular elements from a set. The following statement will choose the element from the set of attributes that has “hblake” as an access Id.

```
user = CHOOSE  x \in attributes: "hblake" \in x["accessid"]
```

These user security attributes will identify which user a thread is acting on behalf of and will flow through the model as requests are carried out.

6.2.4 Access Control Policy and Policy Evaluation

Beyond identifying user security attributes, there are a series of access control related structures that must be modeled: naming resources, guards, dynamic attributes, and policy evaluation. The following sections define the TLA modeling elements used for each of these structures.

6.2.4.1 Resource Naming

In the logic programming model of the previous chapter, resource names were modeled as a list of lists: *[[sr,S],[section,gsi]]*. In the case of TLA, a function will be used for the resource name. For instance, the following function represents the general student information for the student “fblake”. In order to access any particular portion of the name, for example the section requested, the function notation used is: *resource[“section”]*.

```
resource=("sr"       :> "hblake"  @@
          "section" :> "gsi"      @@
          "part"     :>  "all"           )
```

6.2.4.2 Policy Evaluation

The policy evaluator can be modeled in a variety of ways. In this case study model, it is modeled as a single TLA operator which returns true or false after considering a series of logical conditions that represent the access control policy.

```
PolicyEvaluator(a,r,o) == \/ (      /\ r["section"]="gsi"
                                    /\ o="save"
                                    /\ "owner" \in a["roles"]
                              )
                          \/ (  /\ r["section"]="gsi"
                                    /\ o="load"
                                    /\ ( \/ "student" \in a["groups"]
                                         \/ "staff" \in a["groups"]
                                       ) )
```

The preceding policy evaluator operator will give only students the ability to save their own general student information. It will allow any student or staff member to read the general student information for any student record. This form of policy is adequate for this case study, though it is possible in TLA to define custom operators that are defined in Java which can be designed to access some external policy evaluator rather than a model of the policy in TLA. With such a custom operator, it would be possible to perform a model check using the actual access control policy that is to be implemented in the final system rather than using a TLA representation of the policy. This would reduce the likelihood of error being introduced into the model.

6.2.4.3 Guards & Dynamic Attributes

With user security attributes and resource names defined as functions and operations represented with simple strings, it is now possible to define how access control guards can be modeled. As before, the guard needs three pieces of information: user security attributes, resource name, and the operation to be performed. As with the Prolog model, dynamic attributes may be added to the security attributes before the policy is evaluated. The following example guard, from a record server, determines if the user making the request is also the owner of the student record.

```
RS_Guard(a,r,o) == PolicyEvaluator( [ a EXCEPT
                                          !["roles"]=IF (r["sr"] \in a["accessid"])
                                          THEN a["roles"] \cup {"owner"}
                                          ELSE a["roles"]],
                                          r, o )
```

If this is true, then the dynamic attribute *owner* is added to the roles attribute. This is done using the union set operation denoted by *\cup* in TLA. The resulting new attributes, original resource name and operation are then submitted to the policy evaluator operator. The RS_Guard operator will be a guard in the record server protecting access to any particular resource, and the operator can be used in other TLA expressions to determine access authorizations.

6.2.5 Modeling Threads

The TLA model uses two functions to represent threads in the model. First, there is a variable holding a function representing the state of each thread. Threads can be in one of three states: dormant, ready, active. A *dormant* thread is currently available to be assigned to a user request. A thread is *ready* when it has been assigned a user and a request. Once the thread begins processing, its state is changed to *active*. Once the thread has completed work, it reverts back to being dormant. This statement initializes the function, setting all threads to *dormant*, where Threads={“dormant”, “ready”, “active”}.

```
ut_threads = [t \in Threads |-> "dormant"]
```

A second variable is used to hold information concerning the ownership and purpose of each thread. This is represented as a function mapping each thread to a record that holds the user’s security attributes, student access id for the student record being accessed, the action to be taken within the system, and any additional data needed to carry out the request. For instance, the following expression makes thread 1 belong to user “hblake,” who is requesting to load the student record for “acrespi.”

```
ut_threads_data'= [ut_threads_data
            EXCEPT ![1]=  [ r |-> "acrespi",
                              p |-> ("accessid" :> {"hblake"} @@
                                     "roles" :> {} @@
                                     "groups" :> {"student"}),
                              c |-> "load",
                              d |-> [gsi |-> ""]
                 ]
```

The record field *d* will hold the data for the sections of the student record that are loaded from the servers in the system and are initially left blank when the thread is first assigned. Since, in this simplified model, only general student information (gsi) is stored, *d* contains only a result for *gsi*.

6.3 TLA Component Modeling

With this background in TLA and the previous expressions for modeling various aspects of the system and access control functionality, it is now possible to develop component specifications and demonstrate the process of composing the specifications into a system. Each component specification is broken down into six parts: extensions, variables, constants, invariants, initialization, action predicates, and guard operators.

Each component typically has a set of variables that are reserved for use only by that component. The TLA model checker does not support any means of information hiding so even though these variables are only used by a single component, every other component must be aware of these variables so that they can remain unchanged. A component may also have constants for internal use that must be specified. In the case of the user terminal, there are two internal variables, but no local constants.

```
UT_variables: ut_threads, ut_threads_data
UT_constants: N/A
```

Beyond the variables and constants defined for individual components, there will typically be a need for each component to define invariants and to also initialize the values of variables. This is accomplished through two predicates with named prefixes used to avoid name conflicts between components. In the following predicates, *UT_Init* defines *ut_threads* as a function where all threads are all initially dormant, and defines *ut_threads_data* as a function which initially contains no data. The predicate *UT_Inv* validates that *ut_threads* is a member of the set of all possible functions mapping threads to threads states, and *command_set* contains only valid commands.

```
UT_Init ==        /\ ut_threads = [t \in Threads |-> "dormant"]
                  /\ ut_threads_data =[t \in Threads |-> "empty"]

UT_Inv  ==        /\ ut_threads \in [Threads -> ThreadStates]
```

The next step is to define predicates that represent the actions that can take place within a component. As an example, the following four predicates represent the user terminal sending a read student record request to the record server and then receiving the response back from the record server. *UT_Pick_Request(t)* loads a user command from the set of commands into a dormant thread and makes that thread *ready* for processing. *UT_Send_Read(t)* is enabled when the state of the thread is *ready* and the command given to the thread is *read*. If these two conditions are true, then the thread state is changed to *active*, the request to the record server is put in the communications channel *ut_rs_read* and the state of the channel is changed to *sent* to tell the recorder server that data has been sent. Finally, the data representing the thread remains unchanged.

```
UT_Pick_Request(t) ==  /\ ut_threads[t]="dormant"
          /\ Cardinality(command_set)#0
          /\ ut_threads_data'=[ut_threads_data EXCEPT ![t]= CHOOSE x \in
             command_set:x#[p|->("":>""),r|->"",c|->"",d|->""]]
          /\ command_set' = command_set \ {ut_threads_data'[t]}
          /\ ut_threads'=[ut_threads EXCEPT ![t]="ready"]
          /\ UNCHANGED<<principals,attributes >> ]
          /\ UNCHANGED<<ut_rs_save>>


UT_Send_Read(t) ==        /\ ut_threads[t]="ready"
                  /\ ut_threads_data[t].c="read"
                  /\ ut_threads'=[ut_threads EXCEPT ![t]="active"]
                  /\ ut_rs_read[t].state = "ready"
                  /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"sent",
                                                  data|->ut_threads_data[t],
                                                  message |->("":>"")]]
                  /\ UNCHANGED<<ut_threads_data>>
                  /\ UNCHANGED<<command_set, principals,attributes>>
```

```
UT_Receive_Read(t) == /\ ut_threads[t]="active"
                      /\ ut_threads_data[t].c="read"
                      /\ ut_rs_read[t].state = "replied"
                      /\ ut_threads'=[ut_threads EXCEPT ![t]="dormant"]
                      /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"ready",
                                                              data|->ut_threads_data[t],
                                                              message |->("":>"")]]
                      /\ UNCHANGED<< ut_threads_data>>
                      /\ UNCHANGED<<command_set, principals,attributes>>

Waiting == /\ \A t \in Threads: ut_threads[t]="dormant"
           /\ Cardinality(command_set)=0
           /\ UNCHANGED<<ut_threads_data, ut_threads>>
           /\ UNCHANGED<<command_set, principals,attributes>>
```

UT_Receive_Read(t) has the task of waiting for a reply from the record server. For this predicate to be enabled, the thread must be *active*, the command must be *read*, and the channel's state must be *replied*. If these three items are true, then the thread's state is changed to *dormant* and the channel's state is changed to *ready* because the request has been fulfilled. The last predicate, *Waiting*, is enabled when there are dormant threads so that, when there are no user commands, the system will not be in deadlock.

Once all of the predicates representing actions in the component are created, then a single predicate representing the total action of the component is created by logically ORing all of these predicates together. For example, the following expression represents the composition of all of the actions in the user terminal specification.

```
UserTerminal(t) ==      \/ UT_Pick_Request(t)
                        \/ UT_Send_Read(t)
                        \/ UT_Receive_Read(t)
                        \/ Waiting
```

Specifications are thus generated for each concrete component such that their actions are represented in TLA. Lacking from the user terminal specification is any access control related expressions. For an example of the use of a guard operator, consider the following predicate from a record server specification. This action is waiting for requests from a user terminal. Once such a request is presented on the channel, it attempts to read the general student information and reply back to the user terminal.

```
RS_Read(t) ==
    /\ ut_rs_read[t].state = "sent"
    /\ ut_rs_read'=[ut_rs_read
                EXCEPT ![t]=[state|->"replied",
                            data|-> ut_rs_read [t].data,
                         message |-> ( "gsi" :> (IF RS_Guard(ut_rs_read[t].data.p,
                                                         ("sr" :>ut_rs_read[t].data.r @@
                                                          "section" :> "gsi",
                                                          "part" :> "all"          ),
                                                         ut_rs_read[t].data.c)
                                                 THEN gsi[ut_rs_read[t].data.r]
                                                 ELSE ("":>"")  )   )
                            ]
                ]
```

The expression is not enabled until the user terminal has sent the request to the record server. Once this case exists, the state of the channel is updated to indicate that a reply to the user terminal is ready. The guard for the record server is protecting the general student information. If the request is authorized, then the general student information for the requested student is returned, otherwise an empty function is returned indicating no data is available to the user. Because the preceding statement is long and it is not easy examine, the details of the guard operator, the conditional statement with the guard is shown here:

```
IF RS_Guard( ut_rs_read[t].data.p,
      (“sr“ :> ut_rs_read[t].data.r @@ “section“ :> "gsi" @@ "part" :> "all" ),
             ut_rs_read[t].data.c   )
THEN gsi[ut_rs_read[t].data.r]
ELSE ("":>"")
```

The principal information is retrieved from the thread data stored in the communications channel (*ut_rs_read[t].data.p*). The resource name is constructed from the student’s access Id (*ut_rs_read[t].data.r*) and the name of the section being requested. The operation is simply the read command taken from the communications channel (*ut_rs_read[t].data.c*). The record server for this example stores general student information as a function of student access Id in the variable *gsi*. Since the student’s access Id is stored in *ut_rs_read[t].data.r*, then this students general student information is accessed by *gsi[ut_rs_read[t].data.r]*.

Through this process, component TLA models are created by component developers in accordance with standards for the communications channels, variable and predicate names, system wide constants, and access control policy evaluation. With each component providing its own TLA specifications, the system integrator combines these individual specifications together to form the system specification. This process is systematic and outlined in the following section.

6.4 Composing TLA Specifications

When composing TLA specifications, there are several important factors to consider. First, because there is no information hiding, each component's variables add to the state of the entire system. This leads to the following problem. Though a component's predicate can account for unchanging "local" variables, the component developer cannot be aware of the variables another component developer might use in their models. Therefore, when one component's actions are taken, all variables in other components are to remain unchanged. For this to occur, an UNCHANGED<< >> expression must be appended to each action predicate to account for the variables in all other components. This can be done systematically. If done by hand, this process is prone to errors, especially when there are many variables to account for. The second problem is the need to supply user commands to the system for testing purposes. Therefore, the composed specification must include a mechanism for injecting user actions into the system.

To demonstrate the process of composition, the simple two-component system model is used so as to make the process easier to follow. The student information system knowledge-base provides basic information to provide the framework for the specification. The knowledge-base defines the number and location of communications channels, the type of expression used to represent user security attributes, the expressions used to represent user commands submitted to the system via the user terminal, sets that represent the students and staff of the university being modeled, and a global operator for evaluating access control requests. Through this framework, component developers can properly design component TLA models that can interact with system resources and other components. Table 6-5 through Table 6-9 show TLA specifications for this framework and Table 6-10 and Table 6-11 provide the component specifications. Given these portions of the TLA model, the following steps are used to create the composed specifications:

1. Compile a list of all variables used throughout all parts of the specification.
2. Compile a list of all constants used throughout all parts of the specification.
3. Compile a list of all modules that are extended by these specifications.
4. For each component predicate representing an action in the component, add on UNCHANGED<<>> expressions for all of the variables introduced by other components.
5. Concatenate together all of the predicates from all portions of the specification.

Table 6-5: System Constants

```
CONSTANTS Threads,
          ThreadStates,
          Students,
          Staff,
          Command
```

Table 6-6: Channel Related Predicates

```
EXTENDS TLC

VARIABLES ut_rs_read

CH_Init == ut_rs_read = [t \in Threads |-> [state|->"ready",
                                             data |->[p|->("":>""),r|->"",c|->""],
                                             message |->("":>"")]]
```

Table 6-7: Principals, User Attributes and Commands

```
EXTENDS TLC

VARIABLES attributes,
          principals,
          command_set

User_Init == /\ attributes = {( "accessid"  :> {"hblake"}  @@
                                "roles"     :> {} @@
                                "groups"    :> {"student"}  ),
                              ( "accessid"  :> {"fburns"}  @@
                                "roles"     :> {"bursar"} @@
                                "groups"    :> {"staff"}  ),
                              ( "accessid"  :> {"cwinchester"}  @@
                                "roles"     :> {"academicadvisor"} @@
                                "groups"    :> {"staff"}   )    }

             /\ principals = Students \union Staff
             /\ command_set = [p:attributes, c:{"read"}, r:Students,
                               d:{[gsi|->""]}]

User_Inv == /\ principals = Students \union Staff
            /\ command_set \in SUBSET
               [p:attributes, c:Command, r:Students,
                d:{[gsi|->""]}]
```

Table 6-8: Access Control Policy Evaluator Operator

```
PolicyEvaluator(a,r,o) ==\/ (         /\ r["section"]="gsi"
                                      /\ o="save"
                                      /\ "owner" \in a["roles"]
                            )
                          \/ (  /\ r["section"]="gsi"
                                      /\ o="load"
                                      /\ ( \/ "student" \in a["groups"]
                                           \/ "staff" \in a["groups"]
                                         )
                            )
```

Table 6-9: System Level Predicates for Composition

```
TypeInvariant ==  /\ UT_Inv
                  /\ User_Inv

Init  == /\ UT_Init
         /\ RS_Init
         /\ User_Init
         /\ CH_Init

Liveness == WF<< ALL_VARIABLES  >> (NEXT)

Next  ==  \E t \in Threads : ( UserTerminal(t) \/ RecordServer(t) )

Spec  ==  Init /\ □[Next]_<< ALL_VARIABLES  >> /\ Liveness
```

Table 6-10: Example Simplified Record Server TLA Specifications

```
EXTENDS TLC

VARIABLES  gsi

RS_Init == gsi = ( "acrespi" :> << "Alex", "Crespi", "Indianapolis", "IN" >> @@
                   "hblake" :> << "Henry", "Blake", "Washington", "DC" >> @@
                   "ggeorge" :> << "Greg", "George", "Washington", "DC" >> )

RS_Read(t) ==
    /\ ut_rs_read[t].state = "sent"
    /\ ut_rs_read'=[ut_rs_read
                    EXCEPT ![t]=[state|->"replied",
                                 data|-> ut_rs_read [t].data,
                                 message |-> ( "gsi" :> (IF RS_Guard(ut_rs_read[t].data.p,
                                                                     ("sr" :>ut_rs_read[t].data.r @@
                                                                      "section" :> "gsi" @@
                                                                      "part" :> "all"          ),
                                                                     ut_rs_read[t].data.c)
                                                         THEN gsi[ut_rs_read[t].data.r]
                                                         ELSE ("":>"")  )   )
                                 ]
                    ]
    /\ UNCHANGED<<command_set, principals,attributes >>

RS_Guard(a,r,o) == PolicyEvaluator( [ a EXCEPT
                                      !["roles"]=IF (r["sr"] \in a["accessid"])
                                         THEN a["roles"] \cup {"owner"}
                                         ELSE a["roles"]],
                                    r, o )

RecordServer(t) == RS_Read(t)
```

Table 6-11: Example Simplified User Terminal TLA Specifications

```
EXTENDS  TLC,FiniteSets

VARIABLES ut_threads,
                ut_threads_data,

UT_Init == /\ ut_threads = [t \in Threads |-> "dormant"]
             /\ ut_threads_data =[t \in Threads |-> "empty"]

UT_Inv  == /\ ut_threads \in [Threads -> ThreadStates]

UT_Pick_Request(t) ==  /\ ut_threads[t]="dormant"
                       /\ Cardinality(command_set)#0
                       /\ ut_threads_data'=[ut_threads_data EXCEPT ![t]= CHOOSE x \in
                          command_set:x#[p|->("":>""),r|->"",c|->"",d|->""]]
                       /\ command_set' = command_set \ {ut_threads_data'[t]}
                       /\ ut_threads'=[ut_threads EXCEPT ![t]="ready"]
                       /\ UNCHANGED<<principals,attributes >> /\ UNCHANGED<<ut_rs_read>>

UT_Send_Read(t) == /\ ut_threads[t]="ready"
                        /\ ut_threads_data[t].c="read"
                        /\ ut_threads'=[ut_threads EXCEPT ![t]="active"]
                        /\ ut_rs_read[t].state = "ready"
                        /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"sent",
                                          data|->ut_threads_data[t],
                                          message |->("":>"")]]
                        /\ UNCHANGED<<ut_threads_data>>
                        /\ UNCHANGED<<command_set, principals,attributes >>

UT_Receive_Read(t) == /\ ut_threads[t]="active"
                        /\ ut_threads_data[t].c="read"
                        /\ ut_rs_read[t].state = "replied"
                        /\ ut_threads'=[ut_threads EXCEPT ![t]="dormant"]
                        /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"ready",
                                          data|->ut_threads_data[t],
                                          message |->("":>"")]]
                        /\ UNCHANGED<< ut_threads_data>>
                        /\ UNCHANGED<<command_set, principals,attributes >>

Waiting == /\ \A t \in Threads: ut_threads[t]="dormant"
             /\ Cardinality(command_set)=0
             /\ UNCHANGED<<ut_threads_data, ut_threads>>
             /\ UNCHANGED<<command_set, principals,attributes>>

UserTerminal(t) == \/ UT_Pick_Request(t)    \/ UT_Send_Read(t)
                     \/ UT_Receive_Read(t)   \/ Waiting
```

Following the preceding steps, the TLA specification for the simplified student record system is composed. Table 6-12 shows this overall specification with additions highlighted in bold and Table 6-13 shows the configuration file for the specification. Once a model of the system is composed, the system integrator can investigate various properties of the system using the TLC model checker software provided by [LAM03]. The following section examines how such properties are expressed and checked using this software.

Table 6-12: Simplified Student Record System TLA Specification

```
---------------------------------------- MODULE SimpleStudentRecord ----------------------------
EXTENDS FiniteSets, TLC

CONSTANTS Threads, ThreadStates, Students,  Staff, Command

VARIABLES attributes, principals, command_set, ut_rs_read,  ut_threads, ut_threads_data, gsi

CH_Init == ut_rs_read = [t \in Threads |-> [state|->"ready",
                                             data |->[p|->("":>""),r|->"",c|->"", d|->""],
                                             message |->("":>"")]]

User_Init == /\ attributes = {( "accessid" :> {"hblake"}  @@
                                "roles"     :> {} @@
                                "groups"   :> {"student"}  ),
                              ( "accessid":> {"fburns"}  @@
                                "roles"     :> {"bursar"} @@
                                "groups"   :> {"staff"}  ),
                              ( "accessid":> {"cwinchester"}  @@
                                "roles"     :>  {"academicadvisor"} @@
                                "groups"   :> {"staff"}   )     }
             /\ principals = Students \union Staff
             /\ command_set = [p:attributes, c:{"read"}, r:Students, d:{[gsi|->""]}]

User_Inv == /\ principals = Students \union Staff
            /\ command_set \in SUBSET
               [p:attributes, c:Command, r:Students,
                d:{[gsi|->""]}]

PolicyEvaluator(a,r,o) ==\/ (          /\ r["section"]="gsi"
                                       /\ o="save"
                                       /\ "owner" \in a["roles"]  )
                         \/ (  /\ r["section"]="gsi"
                                       /\ o="load"
                                       /\ ( \/ "student" \in a["groups"]
                                            \/ "staff" \in a["groups"]
                                          ) )
```

Table 6-12: continued…

```
UT_Init == /\ ut_threads = [t \in Threads |-> "dormant"]
           /\ ut_threads_data =[t \in Threads |-> "empty"]

UT_Inv  == /\ ut_threads \in [Threads -> ThreadStates]

UT_Pick_Request(t) ==  /\ ut_threads[t]="dormant"
                       /\ Cardinality(command_set)#0
                       /\ ut_threads_data'=[ut_threads_data EXCEPT ![t]= CHOOSE x \in
                          command_set:x#[p|->("":>""),r|->"",c|->"",d|->""]]
                       /\ command_set' = command_set \ {ut_threads_data'[t]}
                       /\ ut_threads'=[ut_threads EXCEPT ![t]="ready"]
                       /\ UNCHANGED<<principals,attributes >>
                       /\ UNCHANGED<<ut_rs_read>>
                       /\UNCHANGED<<gsi>>

UT_Send_Read(t) == /\ ut_threads[t]="ready"
                   /\ ut_threads_data[t].c="read"
                   /\ ut_threads'=[ut_threads EXCEPT ![t]="active"]
                   /\ ut_rs_read[t].state = "ready"
                   /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"sent",
                               data|->ut_threads_data[t],
                               message |->("":>"")]]
                   /\ UNCHANGED<<ut_threads_data>>
                   /\ UNCHANGED<<command_set, principals,attributes >>
                   /\UNCHANGED<<gsi>>

UT_Receive_Read(t) == /\ ut_threads[t]="active"
                   /\ ut_threads_data[t].c="read"
                   /\ ut_rs_read[t].state = "replied"
                   /\ ut_threads'=[ut_threads EXCEPT ![t]="dormant"]
                   /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"ready",
                               data|->ut_threads_data[t],
                               message |->("":>"")]]
                   /\ UNCHANGED<< ut_threads_data>>
                   /\ UNCHANGED<<command_set, principals,attributes >>
                   /\UNCHANGED<<gsi>>

Waiting == /\ \A t \in Threads: ut_threads[t]="dormant"
           /\ Cardinality(command_set)=0
           /\ UNCHANGED<<ut_threads_data, ut_threads>>
           /\ UNCHANGED<<command_set, principals,attributes>>
           /\ UNCHANGED<<gsi>>

UserTerminal(t) == \/ UT_Pick_Request(t)    \/ UT_Send_Read(t)
                   \/ UT_Receive_Read(t)    \/ Waiting

RS_Init == gsi = ( "acrespi" :> << "Alex", "Crespi", "Indianapolis", "IN" >> @@
                   "hblake" :> << "Henry", "Blake", "Washington", "DC" >> @@
                   "ggeorge" :> << "Greg", "George", "Washington", "DC" >> )
```

Table 6-12: continued…

```
RS_Read(t) ==
    /\ ut_rs_read[t].state = "sent"
    /\ ut_rs_read'=[ut_rs_read
                          EXCEPT ![t]=[state|->"replied",
                                               data|-> ut_rs_read [t].data,
                                               message |-> ( "gsi" :> (IF RS_Guard(ut_rs_read[t].data.p,
                                                                                              ("sr" :> ut_rs_read[t].data.r,
                                                                                               "section" :> "gsi",
                                                                                               "part" :> "all"          ),
                                                                                              ut_rs_read[t].data.c)
                                                                            THEN gsi[ut_rs_read[t].data.r]
                                                                            ELSE ("":>"")  )   )
                                                      ]
                         ]
    /\ UNCHANGED<<command_set, principals,attributes,gsi >>
    /\ UNCHANGED<<ut_threads_data, ut_threads>>
    /\ UNCHANGED<<command_set, principals,attributes >>

RS_Guard(a,r,o) == PolicyEvaluator( [ a EXCEPT
                                                       !["roles"]=IF (r["sr"] \in a["accessid"])
                                                           THEN a["roles"] \cup {"owner"}
                                                           ELSE a["roles"]],
                                                      r, o )

RecordServer(t) == RS_Read(t)


TypeInvariant ==  /\ UT_Inv
                         /\ User_Inv

Init  == /\ UT_Init
           /\ RS_Init
           /\ User_Init
           /\ CH_Init

Liveness == WF]_<< attributes, principals, command_set, ut_rs_read,
                              ut_threads, ut_threads_data, gsi >>(NEXT)

Next  ==  \E t \in Threads : ( UserTerminal(t) \/ RecordServer(t) )

Spec  ==  Init /\ [][Next]_<< attributes, principals, command_set, ut_rs_read,
                              ut_threads, ut_threads_data, gsi >> /\ Liveness
===========================================================
```

Table 6-13: Configuration Information for the TLA Model

```
SPECIFICATION Spec

INVARIANT TypeInvariant

CONSTANTS
 Threads    = {1,2}
 Students   = {"acrespi","hblake","ggeorge"}
 Staff      = {"fburns","cwinchster"}
 ThreadStates = {"dormant","ready","active"}
 Command    = {"save","read"}
```

6.5 TLA Modeling Checking

The TLC+ Tools are provided free of charge and without limitation on the user in terms of use, modification, or distribution. These tools are able to perform model checking on specifications including the majority of TLA expressions, though there are some limits in its capabilities, and some TLA specifications cannot be checked in this automatic fashion. For instance, the temporal existential operator, is used in TLA specifications to model information hiding. Though this operator would be useful for modeling components while hiding internal variables, it use is not required as long as variable names are carefully managed so that components do not access each other's variables. The model checker does not support the temporal existential operator. Care has been taken to avoid the use of expressions that the tools are unable to check and such limitations do not adversely affect the ability to create component specifications. This section describes the general use of the TLA model checker as well as methods for testing specification properties.

6.5.1 General Use of the TLC Model Checker

Given a properly formatted TLA specifications and configuration file, the TLC model checker is executed as follows to check the specification in Table 6-12 .

```
Java tlc.TLC SimpleStudentRecord
```

In response, the model checker parses all modules involved in the specification, checks semantics, determines all of the possible initial states, and explores the state space of the model while checking the specified properties. The test shown in Table 6-14 demonstrates that the specification is valid and that the type invariant holds throughout all sequences of actions.

Table 6-14: Example TLC Model Checker Execution

```
TLC Version 2.0 of Jun 10, 2003
Model-checking
Parsing file SimpleStudentRecord.tla
Parsing file C:\tla\tlasany\StandardModules\TLC.tla
Parsing file C:\tla\tlasany\StandardModules\FiniteSets.tla
Parsing file C:\tla\tlasany\StandardModules\Naturals.tla
Parsing file C:\tla\tlasany\StandardModules\Sequences.tla
Semantic processing of module Naturals
Semantic processing of module Sequences
Semantic processing of module TLC
Semantic processing of module FiniteSets
Semantic processing of module SimpleStudentRecord
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
    calculated (optimistic):  2.2113127301492952E-13
    based on the actual fingerprints:  1.2250133276713903E-13
4043 states generated, 2107 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 37.
```

The tester may now test how variations in the access control policy, user credentials, and system data affect the model. Can a change in the policy induce a deadlock in the system? How do changes in user credentials affect the system dynamics? Such tests can be performed on this model before integrating the actual system.

6.5.2 Checking Specification Properties

There are two type general types of properties that are checked using the TLC model checker: safety properties and liveness properties. Safety properties only specify what cannot happen. For instance, the specification must satisfy the invariants being tested and the next predicate must always be enabled. Although these are important properties for specifications, safety properties do not determine what must happen. In the case of the simple student record specification, is it required to even process a request to completion or can there be starvation? Such properties are termed liveness properties and they can be specified and checked. For instance, one liveness property to check is to determine if every message sent to the record server eventually is replied to. Currently the example specification does not check for any liveness properties. In order to check the liveness property for the channel, add an expression for a liveness property in the specification to determine if when that channel enters the *sent* state that eventually (~>) it will enter the *replied* state.

```
Channel_works== \A t \in Threads : (ut_rs_read[t].state = "sent") ~>
                                    (ut_rs_read[t].state = "replied")
```

Then add the line property *Channel_works* to the configuration file. This will result in requesting that the model checker validate the model for this property for all behaviors in the system as shown in Table 6-15. Additional liveness properties can be added in the same fashion to examine aspects of the system's dynamics.

Table 6-15: Checking Temporal Properties

```
TLC Version 2.0 of Jun 10, 2003
Model-checking
Parsing file SimpleStudentRecord.tla
Parsing file C:\tla\tlasany\StandardModules\TLC.tla
Parsing file C:\tla\tlasany\StandardModules\FiniteSets.tla
Parsing file C:\tla\tlasany\StandardModules\Naturals.tla
Parsing file C:\tla\tlasany\StandardModules\Sequences.tla
Semantic processing of module Naturals
Semantic processing of module Sequences
Semantic processing of module TLC
Semantic processing of module FiniteSets
Semantic processing of module SimpleStudentRecord
Implied-temporal checking--satisfiability problem has 2 branches.
Finished computing initial states: 1 distinct state generated.
--Checking temporal properties for the complete state space...
Model checking completed. No error has been found.
  Estimates of the probability that TLC did not check all reachable states
  because two distinct states had the same fingerprint:
    calculated (optimistic):  2.2113127301492952E-13
    based on the actual fingerprints:  1.2250133276713903E-13
4043 states generated, 2107 distinct states found, 0 states left on queue.
The depth of the complete state graph search is 37.
```

Temporal logic can also be used to validate behaviors in the specifications with respect to the access control policy. For instance, students should always be able to ready their own general student information. This constraint can be expressed in the following TLA predicate:

```
Read_self == \A t \in Threads :( (/\ ENABLED(UT_Send_Read(t))
                          /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                          /\ ut_threads_data[t].c="read")
                         ~>
                         (/\ ENABLED(UT_Receive_Read(t))
                          /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                          /\ ut_threads_data[t].c="read"
                          /\ ut_rs_read[t].message#("":>"")) )
```

In this expression, if *UT_Send_Read(t)* is enabled and the request is to read the users' own student record, then eventually *UT_Receive_Read(t)* will be enabled and the message in the channel will not be empty indicating that the information was released to the user. Using expressions of this type, it is possible to validate the behavior of the system with respect to the access control policies being used. The results of such testing should correlate with the logic programming model testing results using the techniques from the previous chapter.

This chapter explores the modeling of components and system using the Temporal Logic of Actions. Althrough the techniques introduced, it is possible to validate the composed TLA specifications to be deadlock free and possess defined liveness properties. This will provide the system integrator with information used to statically validate a system composition before deployment as well as allow for testing the impact of access control policies on the functionality of the system. Chapter 7 combines the techniques from Chapters 4, 5, and 6 into a full case study that tests the search strategies as well as the implements logic programming and TLA models. The properties of the modeled systems are then compared to system implementations using java rmi-based distributed components.

7. CASE STUDY EXPERIMENTATION

This chapter examines the ideas presented in this thesis in the context of the student information system case study introduced in Chapter 3. Through this case study, component search strategies will be tested and predications made about system compositions based on the logic programming and TLA+ will be compared to testing of several systems composed of a variety of components. Section 7.1 outlines the general structure of the system being tested. Section 7.2 presents various components used in the testing and how their specifications are written. Section 7.3 demonstrates component searching using a simulation of a headhunter. Section 7.4 outlines the four system variations that were created and testing in the case study. Section 7.5 describes the results from logic program access control modeling for each of these four systems. Section 7.6 presents the TLA+ access control modeling for each of the four systems. Finally, Section 7.7 presents the actual runtime results collected from the test systems and correlates these results with the model predications.

7.1 Student Information System

Although this case study uses the basic student information system outlined in Chapter 3, one modification is made to the system model to facilitate the ease of implementing access control policies for all components in the system. In this case study, each of the server components has access to a *SecurityCenter* component which does the task of evaluating any access control request from any component in the system and returning a yes or no result to the component. The choice of a single security center is not crucial for the system. It would be possible for each component to have its own security center or for groups of components to share security centers. Such variations would need to be described in the UniFrame generative domain model for the student information system and selected by the system integrator. The centralized security center model would be one possible system in the generative domain model. There will be four resources enclosed in the system: general student information (gsi), academic records (ar), financial aid records (fa), and student employment records (se).

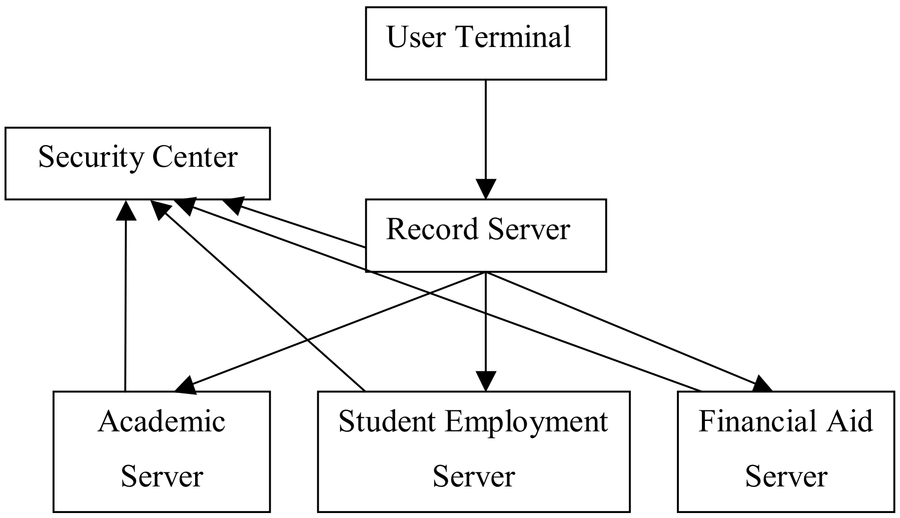

Figure 7-1: Student Information System

7.2 Component Design and Implementation

Twelve different components were specified, developed, and composed into four variations of the system shown in Figure 7-1. Two component types *UserTerminal* and *SecurityCenter* have only one concrete implementation each. The component types *AcademicRecordServer*, *FinancialAidServer*, and *StudentEmploymentServer* each have two concrete implementations each and the *RecordServer* component type has four concrete implementations. The variations in the access control properties of these components are outlined in Table 7-1. For example, *AcademicRecordServer1* contains academic records (*.ar) but does not provide a way of protecting that resource. Two components deserve special explanations. *RecordServer3* and *RecordServer4* are implementations of the *RecordServer* abstract component that have internal caches for caching information from the lower level servers. *RecordServer3* has no mechanism for guarding access to this cached information whereas *RecordServer4* guards access to these caches. These two components will be used to demonstrate how the TLA+ modeling can identify access control problems created by the inappropriate application of *RecordServer3*.

Table 7-1: Concrete Component Access Control Specifications

Abstract Component	Concrete Component	Contained Resources	Protected Resources
SecurityCenter	SecurityCenter	NONE	NONE
UserTerminal	UserTerminal1	NONE	NONE
RecordServer	RecordServer1	*.gsi	NONE
	RecordServer2	*.gsi	*.gsi
	RecordServer3	*.gsi, *.ar, *.fa, *.se	*.gsi
	RecordServer4	*.gsi, *.ar, *.fa, *.se	*.gsi, *.ar, *.fa, *.se
AcademicRecordServer	AcademicRecordServer1	*.ar	NONE
	AcademicRecordServer2	*.ar	*.ar
FinancialAidServer	FinancialAidServer1	*.fa	NONE
	FinancialAidServer2	*.fa	*.fa
StudentEmploymentServer	StudentEmploymentServer1	*.se	NONE
	StudentEmploymentServer2	*.se	*.se

7.3 Component Searching

Component searching was tested in the context of a single headhunter. In the URDS, the Query Manager propagates a query to headhunters who then perform the actually component matching. Therefore, existing headhunters need only be modified to filter component matches through an additional step that filters out components that do not meet access control specifications. Since testing the access control matching algorithm only involves changes to the headhunter, a simulation was created in which a headhunter loads concrete component UMM specifications directly from XML files rather than from an active registry. The headhunter then creates its meta-repository from these XML specifications. The implementation of this simulation is provided in Appendix G.

The simulation instantiates a headhunter and then submits, to the headhunter, a series of queries, one for each abstract component type needed create the system. One component of each type is chosen from the search results and the simulation then proceeds to automatically create the Prolog access control model for the system using the specifications for each individual component. Appendix A provides listings of the UMM concrete component specifications in XML format for each concrete component. Appendix B provides UMM abstract component specifications used to submit queries. Some examples from testing the search algorithm on components are listed in Table 7-3. For each query submitted to the headhunter, it creates a Prolog matching program from the query specifications and from the access control specifications for each component to be considered. The process of creating this matching program was described in Chapter 4 of this thesis and Table 7-2 shows the matching program generated for analyzing the *StudentEmploymentServer* specifications for Query 3. In this case, because the student employment resource does not require protection, the matching program returns both *StudentEmploymentServer1* and *StudentEmploymentServer2.* Each test performed on the matching algorithm successfully identified components with access control specifications matching the query's requirements.

7.4 System Variations

Given four choices for record servers and two choices for each of the other servers, there are a variety of system configurations that can be tested. Four such systems were examined in detail in this case study. Three cases represent the composition of systems that successfully meet the system requirements for access control properties. The fourth case represents the composition of a system that will not satisfy the system requirements. Although this fourth case should not be allowed based on the search result, it is presented to show how the Prolog and TLA models can successfully verify system behavior or in this case identify improper system behavior. The problem in the fourth system results from the read results, cached from one user's request, being used to fulfill another user's request. Since *RecordServer3*'s caches have no access control protection, once an authorized user accesses a record from a lower level server, any other user may read the record even if not properly authorized. Although this situation should not be allowed if the headhunter's search results are properly used, it is included to demonstrate how the Prolog and TLA+ access control models can be used to identify such problems before system composition.

Table 7-2: Query 3 *StudentEmploymentServer* Prolog Matching Program

```
require_protect(['gsi','ar','fa']).
consider('StudentEmploymentServer1').
consider('StudentEmploymentServer2').


contain('StudentEmploymentServer1','se').
ensure_protect('StudentEmploymentServer1','').
contain('StudentEmploymentServer2','se').
ensure_protect('StudentEmploymentServer2','se').


protects(_,[]).

protects(Component,[Resource|OtherResources]) :-
   (contain(Component,Resource),ensure_protect(Component,Resource) ;
   not contain (Component,Resource)),
   protects(Component,OtherResources).


match(Component) :- consider(Component),
                 require_protect(Resources),
                 protects(Component,Resources).
```

Table 7-3: Component Search Specifications and Results

Query	Resources Requiring Protection	Components Returned
1	NONE	UserTerminal1 RecordServer1 RecordServer2 RecordServer3 RecordServer4 AcademicRecordServer1 AcademicRecordServer2 FinancialAidServer1 FinancialAidServer2 StudentEmploymentServer1 StudentEmploymentServer2
2	*.gsi, *.ar, *.fa, *.se	UserTerminal1 RecordServer2 RecordServer4 AcademicRecordServer2 FinancialAidServer2 StudentEmploymentServer2
3	*.gsi, *.ar, *.fa	UserTerminal1 RecordServer2 RecordServer4 AcademicRecordServer2 FinancialAidServer2 StudentEmploymentServer1 StudentEmploymentServer2

7.5 Logic Programming Access Control Modeling Results

As mentioned previously, the search simulation was designed to automatically create a Prolog-base access control model for each system composed from selected components returned by the headhunter. This process is outlined in Chapter 5. For each of the four systems listed in Table 7-4, the Prolog access control model was generated and tested. Appendix C provides a listing for the model of system 2. A series of queries were presented to each model which should produce results in accordance with the access control policy stored in the security center and the access control properties of the system components. The following sections examine the results of these queries for each test system.

Table 7-4: Four Tested System Compositions

System	Expected Protected Resources	Components Used
1	NONE	UserTerminal1 RecordServer1 AcademicRecordServer1 FinancialAidServer1 StudentEmploymentServer1
2	*.gsi, *.ar, *.fa, *.se	UserTerminal1 RecordServer2 AcademicRecordServer2 FinancialAidServer2 StudentEmploymentServer2

Table 7-4: continued…

System	Expected Protected Resources	Components Used
3	*.gsi, *.ar, *.fa, *.se	UserTerminal1 RecordServer4 AcademicRecordServer2 FinancialAidServer2 StudentEmploymentServer2
4	*.gsi, *.ar, *.fa, *.se	UserTerminal1 RecordServer3** AcademicRecordServer2 FinancialAidServer2 StudentEmploymentServer2

** *RecordServer3* will introduce access control problems when reading records.

7.5.1 Querying of System 1 Prolog Model

Since this system is composed from components with no access control guards, any user will be able to perform any operation on the system. For instance the following query verifies that each of the possible five users can read each of the two student's records in the system. It returns every combination of user (U) and student (S) that results in a 'yes' to the query and shows that nobody is blocked from reading any student record.

```
| ?- user_terminal_readStudentRecord(U,S).
U = acrespi
S = acrespi?;
U = acrespi
S = hblake?;
U = hblake
S = acrespi?;
U = hblake
S = hblake?;
```

```
U = fburns
S = acrespi?;
U = fburns
S = hblake?;
U = cwinchester
S = acrespi?;
U = cwinchester
S = hblake?;
```

Verifying the saving operation behavior take more work because of the different combinations of sections of the student record that can be written. For instance, the following query verifies which users are allowed to save to all sections of different student records. As expected, anyone can save to any section of any student record.

```
| ?- user_terminal_saveStudentRecord(U,S,[gsi,as,fa,se]).
U = acrespi
S = acrespi?;
U = acrespi
S = hblake?;
U = hblake
S = acrespi?;
U = hblake
S = hblake?;
U = fburns
S = acrespi?;
U = fburns
S = hblake?;
U = cwinchester
S = acrespi?;
U = cwinchester
S = hblake?;
```

For the first system, these two queries are sufficient to show that no resources are protected by the system. Additional queries will be required to test the remaining three system models.

7.5.2 Querying of System 2 Prolog Model

The second system consists of components that regulate access to all resources and therefore the access control policy in the *SecurityCenter* will be authorizing access to the resources. The first query from the previous section is submitted to this next system model. The model provides output for each access control request made during a query but this output is removed from the following listing in order to highlight the overall results of the query. In this case, students (acrespi and hblake) can only read their own record. The bursar (fburns) and the academic advisor (cwinchester) can access any student record. This illustrates one limitation of this Prolog model for reading. Even though some users can successfully read student records, the model does not indicate which sections of the student record are read or blocked from being read. A successful read only indicates that at least one section of the student record is readable by the user.

```
| ?- user_terminal_readStudentRecord(U,S).
U = acrespi
S = acrespi?;
U = hblake
S = hblake?;
U = fburns
S = acrespi?;
U = fburns
S = hblake?;
U = cwinchester
S = acrespi?;
U = cwinchester
S = hblake?;
```

Submitting the query to save all sections of a student record results in the action being blocked for all users. This is expected as students are only allowed to save their own general student information, bursars can only save financial aid information and student employment information, and academic advisors can only save academic records. No user can save all sections of a student record so this result is expected.

```
| ?- user_terminal_saveStudentRecord(U,S,[gsi,as,fa,se]).
no
```

The next queries determine that students can only save their own general student information. The results are as expected from the access control policy.

```
| ?- user_terminal_saveStudentRecord(S,S,[gsi]).
S = acrespi?;
S = hblake?;
```

```
| ?- user_terminal_saveStudentRecord(S,S,[ar]).
no

| ?- user_terminal_saveStudentRecord(S,S,[fa]).
no

| ?- user_terminal_saveStudentRecord(S,S,[se]).
no
```

7.5.3 Querying of System 3 Prolog Model

The third system consists of components that regulate access to all resources and therefore the access control policy in the *SecurityCenter* will be authorizing access to the resources. In this case, the *RecordServer4* component not only contains the general student information, but also contains caches for the other sections of the student record. These caches are protected with access control guards, so it is expected that this system will have the same access control behavior as the previous system. As expected, these query results match those from the previous system.

```
| ?- user_terminal_readStudentRecord(U,S).
U = acrespi
S = acrespi?;
U = hblake
S = hblake?;
U = fburns
S = acrespi?;
U = fburns
S = hblake?;
U = cwinchester
S = acrespi?;
U = cwinchester
S = hblake?;

| ?- user_terminal_saveStudentRecord(S,S,[gsi]).
S = acrespi?;
S = hblake?;

| ?- user_terminal_saveStudentRecord(S,S,[ar]).
no
| ?- user_terminal_saveStudentRecord(S,S,[fa]).
no
| ?- user_terminal_saveStudentRecord(S,S,[se]).
no
```

7.5.4 Querying of System 4 Prolog Model

The fourth system consists of components that regulate access to all resources and therefore the access control policy in the *SecurityCenter* will be authorizing access to the resources. In this case, the *RecordServer3* component not only contains the general student information, but also contains caches for the other sections of the student record. In this case though, the caches are not protected with access control guards. This should allow students to read other students' records in some cases. Testing this system will require modifying the Prolog code to investigate the effect of the cache contents on the success of reading. First, the cache is modeled as empty by changing the cache predicates to contain meaningless information:

```
record_server_ar_cache(e,e,e).
record_server_fa_cache(e,e,e,e,e).
record_server_se_cache(e,e,e).
```

Now the same queries are again submitted to the model. It is expected, that with empty caches, the student record information will still be protected properly through the guards in the lower level servers. As the results below show, the system behaves as expected.

```
| ?- user_terminal_readStudentRecord(U,S).
U = acrespi
S = acrespi?;
U = hblake
S = hblake?;
U = fburns
S = acrespi?;
U = fburns
S = hblake?;
U = cwinchester
S = acrespi?;
U = cwinchester
S = hblake?;

| ?- user_terminal_saveStudentRecord(S,S,[gsi]).
S = acrespi?;
S = hblake?;

| ?- user_terminal_saveStudentRecord(S,S,[ar]).
no
```

```
| ?- user_terminal_saveStudentRecord(S,S,[fa]).
no
| ?- user_terminal_saveStudentRecord(S,S,[se]).
no
```

Next, the cache predicates are set to contain information for a student and the same queries are tested again. In this case, record information for the student hblake is added to the cache. It is now expected that student acrespi will now be able to read hblake's student record because *RecordServer3*'s caches are not guarded and the read request will never reach the lower level servers. As expected, acrespi is now able to read hblake's student record. The remaining saving queries remain unchanged.

```
record_server_ar_cache(hblake,dps([chemistry]),ts([[c101,a],[c132,a]])).
record_server_fa_cache(hblake,[bankofamerica,sallemay],[pell],[volleyball],[]).
record_server_se_cache(hblake,[[fall2003],[spring2004],[fall2004]],[]).

| ?- user_terminal_readStudentRecord(U,S).
U = acrespi
S = acrespi?;
U = acrespi
S = hblake?;
U = hblake
S = hblake?;
U = fburns
S = acrespi?;
U = fburns
S = hblake?;
U = cwinchester
S = acrespi?;
U = cwinchester
S = hblake?;

| ?- user_terminal_saveStudentRecord(S,S,[gsi]).
S = acrespi?;
S = hblake?;


| ?- user_terminal_saveStudentRecord(S,S,[ar]).
no
| ?- user_terminal_saveStudentRecord(S,S,[fa]).
no
| ?- user_terminal_saveStudentRecord(S,S,[se]).
no
```

These results show that this Prolog-based access control modeling can predict which system level functions a user can properly execute. There are two difficulties that become apparent from this case study. First, investigating the model's properties may involve modifying the model's representation of data as was done in the cache example. In such cases, it may be easy to overlook possibilities that may introduce access control problems. Second, though the queries can determine if a function can be executed or not, it currently does predict to what extent a system function is carried out. This is evident in the reading student record examples where there is no information concerning what portions of the student record were indeed read. Therefore, this model is unable to determine the degree of filtering done to the student record before being presented to the user. The save functionality, which is all or nothing, only succeeds if all sections presented to be saved are in fact saved. This allows for save function to be investigated at a finer degree of detail.

7.6 TLA+ Access Control Modeling Results

Beyond to the preceding Prolog modeling, TLA+ models are used to investigate additional system properties. As described in Chapter 6, the TLC model checker will check for safety properties, deadlock, and liveness properties. Each of the system components is modeled via TLA+ and these component models are presented in Appendix D. Following the process outlined in Chapter 6, the component specifications are composed into four different system specifications. System 2's specifications are listed in Appendix E. These four system specifications are then easily checked for safety properties and deadlock through the use of the model checking software. All four system models resulted in deadlock free systems that broke no safety properties such as invariants. With these basic tests completed, the modeling checking can now move into testing liveness properties which can verify the access control behavior of the system.

7.6.1 TLA+ Liveness Properties for Access Control Verification

A series of temporal expressions are added to each TLA+ model to test various cases in terms of who can do what in the system. For instance, a student should be able to read all sections of their own student record. This property is tested through the following TLA expressions.

```
Start_Read_Self(t) == /\ ENABLED(UT_Send_Read(t))
                    /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]

End_Read_Self(t) == /\ ENABLED(UT_Receive_Read(t))
                  /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                  /\ ut_rs_read[t].message["gsi"]="READ"
                  /\ ut_rs_read[t].message["ar"] ="READ"
                  /\ ut_rs_read[t].message["fa"] ="READ"
                  /\ ut_rs_read[t].message["se"] ="READ"

Self_Read ==\A t \in Threads: Start_Read_Self(t) ~> End_Read_Self(t)
```

The first predicate is true when the *UT_Send_Read(t)* predicate is enabled and Id of the student record being accessed (*ut_threads_data[t].r*) is the same as the user's accessid (*ut_threads_data[t].p["accessid"]*). This should be true when a read command is first loaded into the system. The second predicate is true when the user terminal receives a response back from the record server, the user is the student access their own record, and all section are read properly. The third expression uses the "leads to" operator ~> to make sure that if *Start_Read_Self(t)* is true then *End_Read_Self(t)* will eventually be true during that behavior. If during model checking, all possible system behaviors result in *Self_Read* being true, then the model does allow students to always read all portions of their own student record. If for some reason, one or more of the sections is not read properly, this temporal logic expression will be false and the model checker will report the model failure. There are several other temporal properties that can be easily verified.

- No student should be able to read any portion of another student's record.
- No student should be able to save any portion of another student's record.
- A student can only save to their own general student information.

TLA specifications to test each of these conditions are shown in Table 7-5, Table 7-6, and Table 7-7 respectively. These by no means represent an exhaustive list of access control properties that can be verified but instead are aimed to demonstrate the ability of TLA+ to perform these tests before a system is composed and deployed or before access control policies are updated in a live system. These predicates are added to each system specification and the configuration file is updated to check these additional properties as listed in Table 7-8. Each of the four system models is then model checked to verify these additional properties.

Table 7-5: Cannot Read Another Student’s Record

```
Start_Read_Other_Student(t) == /\ ENABLED(UT_Send_Read(t))
                  /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                  /\ "student" \in ut_threads_data[t].p["groups"]


End_Read_Other_Student(t) == /\ ENABLED(UT_Receive_Read(t))
                 /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                 /\ "student" \in ut_threads_data[t].p["groups"]
                 /\ ut_rs_read[t].message["gsi"]="BLOCKED"
                 /\ ut_rs_read[t].message["ar"] ="BLOCKED"
                 /\ ut_rs_read[t].message["fa"] ="BLOCKED"
                 /\ ut_rs_read[t].message["se"] ="BLOCKED"


Other_Student_Read ==\A t \in Threads: Start_Read_Other_Student(t) ~> End_Read_Other_Student(t)
```

Table 7-6: Cannot Save Another Student’s Record

```
Start_Save_Other_Student(t) == /\ ENABLED(UT_Send_Save(t))
                  /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                  /\ ut_threads_data[t].d # [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]
                  /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Other_Student(t) == /\ ENABLED(UT_Receive_Save(t))
                   /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                   /\ ut_threads_data[t].d # [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]
                   /\ "student" \in ut_threads_data[t].p["groups"]
                   /\ ut_rs_save[t].message="BLOCKED"

Other_Student_Save ==\A t \in Threads: Start_Save_Other_Student(t) ~> End_Save_Other_Student(t)
```

Table 7-7: A Student Saves Only to Own Record

```
Start_Save_Self_GSI(t) == /\ ENABLED(UT_Send_Save(t))
                  /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                  /\ ut_threads_data[t].d = [gsi |->"y", ar |->"n", fa |->"n", se |->"n"]
                  /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Self_GSI(t) == /\ ENABLED(UT_Receive_Save(t))
                    /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                    /\ ut_threads_data[t].d = [gsi |->"y", ar |->"n", fa |->"n", se |->"n"]
                    /\ "student" \in ut_threads_data[t].p["groups"]
                    /\ ut_rs_save[t].message="SAVED"

Start_Save_Self_notGSI(t) == /\ ENABLED(UT_Send_Save(t))
                  /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                  /\ ut_threads_data[t].d \in [gsi:{"y","n"}, ar:{"y","n"}, fa:{"y","n"}, se:{"y","n"}]
                                   \ {[gsi |->"y", ar |->"n", fa |->"n", se |->"n"],
                                     [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]}
                  /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Self_notGSI(t) == /\ ENABLED(UT_Receive_Save(t))
                  /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                  /\ ut_threads_data[t].d \in [gsi:{"y","n"}, ar:{"y","n"}, fa:{"y","n"}, se:{"y","n"}]
                                   \ {[gsi |->"y", ar |->"n", fa |->"n", se |->"n"],
                                     [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]}
                  /\ "student" \in ut_threads_data[t].p["groups"]
                  /\ ut_rs_save[t].message="BLOCKED"

Self_Save ==\A t \in Threads: /\ (Start_Save_Self_GSI(t) ~> End_Save_Self_GSI(t))
                              /\ (Start_Save_Self_notGSI(t) ~> End_Save_Self_notGSI(t))
```

Table 7-8: Configuration File to Check Temporal Properties

```
CONSTANTS
  Threads    = {1,2}
  Students   = {"acrespi","hblake"}
  Staff      = {"fburns","cwinchster"}
  UTCommand    = {"save","read"}
  ThreadStates = {"dormant","ready","active"}

SPECIFICATION Spec

INVARIANT TypeInvariant

PROPERTY Self_Read Other_Student_Read Other_Student_Save Self_Save
```

7.6.2 Model Checking Results for Four System Variations

Table 7-9 summarizes the results of these model checking attempts. As expected, systems two and three succeed on all accounts because the access control policy is properly implemented by the specification. The first System cannot possible satisfy these constraints because there are not guards in the system. The fourth system fails because the unprotected caches allow students to potentially read other student's records.

Table 7-9: TLA+ Model Checking Results for Case Study Systems

System	Result
1	Failure: These conditions are false. • No student should be able to read any portion of another student's record. • No student should be able to save any portion of another student's record. • A student can only save to their own general student information.
2	Success
3	Success
4	Failure: This condition is false. • No student should be able to read any portion of another student's record.

7.7 Runtime Results and Comparison with Models

Of course, all of the previous modeling results are meaningless without a comparison with an actual system. This section summarizes the results from test runs on these four systems using concrete components. All components are created using Java and communicate via Remote Method Invocation. After each system is composed, a testing program submits each possible read and save command to the system and tabulates the results for each test. In addition, the system logs the behavior of each component in the system. The logs for each component are combined into a trace of the entire system's behavior during the test run. This log provides a detailed view of the system behavior. The tabulated results for each operation on each system are presented in Appendix H.

As expected, the first system successfully allowed all accesses to all student records. All read and all save operations were successfully and all portions of all records are allowed to be read. The second and third systems again produced identical results which are summarized below. Only successful save operations are listed in these results in order keep listing to a reasonable size. As Table 7-10 indicates, the system allows students to read all portions of their own student record while blocks their access to all portions of other students' records. The bursar (*fburns*) can read any general student information, financial aid data, and student employment records. Also, the academic advisor (*cwinchster*) can access both general student information and academic records.

The YES/NO entries in Table 7-11 indicate if that section of the student record is trying to be saved in that attempt. For instance, the entry for *fburns* involves only trying to saved the student employment information indicated by the YES in the SE column. These save results also agree with the models. Students can only save to their own general student information section. The bursar can only save financial aid and/or student employment records. Finally, the academic advisor can only save academic records. These results are all predicated by the Prolog and TLA+ models indicating that the models can predict the system behavior in terms of access control related properties. There is one more case to verify: does the forth system fails to protect reading of the academic records, financial aid records, and student employment records?

The failure of the fourth system is easily seen in the summary of results for reading student records. In Table 7-12, since the student acrespi successfully reads all portions of his own student record, all subsequent read accesses for that record are successful even though the lower level components have access control guards. Therefore, the caches are indeed allowing improper resource accesses as predicted by both the Prolog and TLA+ models.

Table 7-10: Summary of Read Results for Systems Two and Three

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE
acrespi	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
acrespi	READ	hblake	BLOCKED	BLOCKED	BLOCKED	BLOCKED
hblake	READ	acrespi	BLOCKED	BLOCKED	BLOCKED	BLOCKED
Hblake	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
Fburns	READ	acrespi	ACCESSED	BLOCKED	ACCESSED	ACCESSED
Fburns	READ	hblake	ACCESSED	BLOCKED	ACCESSED	ACCESSED
cwinchester	READ	acrespi	ACCESSED	ACCESSED	BLOCKED	BLOCKED
cwinchester	READ	hblake	ACCESSED	ACCESSED	BLOCKED	BLOCKED

Table 7-11: Summary of Successful Save Results for Systems Two and Three

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE	RESULT
acrespi	SAVE	acrespi	YES	NO	NO	NO	SUCCEEDED
hblake	SAVE	hblake	YES	NO	NO	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	NO	YES	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	YES	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	YES	YES	SUCCEEDED
fburns	SAVE	hblake	NO	NO	NO	YES	SUCCEEDED
fburns	SAVE	hblake	NO	NO	YES	NO	SUCCEEDED
fburns	SAVE	hblake	NO	NO	YES	YES	SUCCEEDED
cwinchester	SAVE	acrespi	NO	YES	NO	NO	SUCCEEDED
cwinchester	SAVE	hblake	NO	YES	NO	NO	SUCCEEDED

Table 7-12: Summary of Reading Results for System Four

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE
acrespi	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
acrespi	READ	hblake	BLOCKED	ACCESSED	ACCESSED	ACCESSED
hblake	READ	acrespi	BLOCKED	ACCESSED	ACCESSED	ACCESSED
hblake	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
fburns	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
Fburns	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
cwinchester	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
cwinchester	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED

This chapter investigates and verifies the access control properties of four example student record systems. These models allow for proposed systems to be tested for proper behaviors with regard to access control policies. Therefore, before a system is composed, the system integrator can have some level of assurance that the system will act as predicted. In addition, changes in access control policies can be verified before they are applied to an existing system. With this level of verification, the system integrator can be more assured of the security properties of the system related to access control. The next chapter summarizes the ideas of this thesis, draws conclusions from the case study, and presents possible avenues for future work.

8. CONCLUSION

This thesis presented an access control model for the UniFrame framework for the purpose of presenting access control as a composable quality of distributed software components. Section 8.1 presents the objectives and contributions of this research. Section 8.2 discusses possible future work related to this research and Section 8.3 provides a summary of this thesis.

8.1 Outcome of the Study

Software solutions for distributed computing systems require means for specifying, predicting, and verifying system properties with respect to the properties of individual components. This thesis presents an access control model for UniFrame-based systems that allows for the specification and verification of access control properties of a DCS. This thesis extends the UniFrame Approach to automated assembly of DCS from components to include access control specifications. The specific objectives of this thesis were:

- To provide a means to express the access control characteristics of distributed computing systems in the UniFrame paradigm.
- To provide a means of expressing the access control characteristics of individual software components.
- To create a means of identifying individual software components that meet system access control requirements.
- To provide a means of statically predicting the access control behavior of an integrated system based on the properties of individual components.

In order to facilitate the automated assembly of distributed component-based computing systems, this thesis defines a means of characterizing, decomposing, and composing the access control security characteristics of distributed software components and the systems integrated from these components. The following list outlines the contributions of this thesis.

- Provides an access control framework for the UniFrame Approach.
    - Extended the UMM Specification for access control properties.
    - Created a matching algorithm for extending Resource Discovery to include access control properties.
    - Extended Knowledge-Base to include access control related information.
        - Resource definition and naming.
        - TLA system model framework.
        - Prolog system model framework.
        - Predicate naming conventions.
- Static verification of access control policy implementation using TLA.
- Prediction of effects of access control policy on system dynamics.
- Prediction of effects of component access control properties on system behavior.
- Evaluate changes in access control policy before implementation.
- Prediction of user access privileges to system level functionality.

The TLA+ modeling provided useful predictions due to its ability test a wide variety of system properties in order to help the system integrator be more assured of the impact of access control policies on the functionality of the system for a variety of users. The Prolog-based modeling, although potentially useful in systems where access control decisions are not based on dynamic attributes, is limited in usefulness when the tester must manually explore how changes to data in the model affect the system's behavior. The TLA+ modeling can be designed to automatically explore these options. The major limitation of TLA+ is the shear size of the state space that might need to be explored to test a model. As more and more complex systems are tested, the state space grows very quickly and can soon overwhelm a computer's ability to find all reachable states. Therefore, TLA+ component models should be kept as simple as possible while still properly modeling the behavior of the actually concrete component it represents. There is a trade off that the more simplified the model is then the less assured a tester will be of the properties of a component or system. Slow performance in the model checking may be worthwhile in cases where higher levels of assurance are needed and some delay before system integration is tolerable. On the other extreme, if model checking is eliminated and solely the matching results are used to predict system behavior, then the system integrator can still be assured that the proper resources can be protected. However, there will be limited assurance that the access control policy will be properly implemented and no assurance that the access control properties will not adversely affect the system dynamics such as causing a deadlock.

8.2 Future Work

There are several areas where the model proposed in this thesis can be extended in the future work. The generation of the system TLA+ specifications is not yet automated. The process of composing this model is systematic and should lend itself to being automated, and a tool for composing the system TLA+ specifications would speed the model checking process. The Prolog-based access control modeling should be modified to handle access control-based filtering of results, such as during the reading of student records, better. The current Prolog model is of a limited use in such situations because it gives no indication of what information is released. It may be possible to develop more comprehensive Prolog models that can accomplish this task. Next, the access control modeling should be fully integrated into the existing UniFrame System Generator and URDS systems. Future work should also investigate the effects of component heterogeneity on this access control framework. The heterogeneity may be technological, syntactic, and semantic and will affect how components interact in a system. Will this modeling holdup in heterogeneous systems, or are there modifications that will be necessary? Beyond the access control, additional security properties should be investigated and added to the UniFrame Framework. These security properties could include, but are not limited to, encryption and authentication. It may prove useful to try to extend the model checking process into a method of evaluating an integrated system at runtime. It may be possible to then monitor system activity such that access control problems can be identified at runtime or as a way to predict success or failure of an operation before making a request of a system. Another avenue for future work is extending the TLC model checker to give more verbose feedback when properties are violated. If many properties are being check simultaneously and one is violated, the model checker is unable to indicate which property was violated and the work of isolating the problem is left up to the tester. Finally, this thesis does not explore the possibility of inserted guards between components in an integrated system. How fine grained of control can be gained through such guards and how will they affect the system dynamics?

8.3 Summary

This thesis has presented an access control model for the UniFrame Approach which facilitates the inclusion of access control properties into the semi-automated process of generating distributed computing systems developed in previous research. Initially inspired by the work of [BUR03] on model driven access control, this thesis applies the five principles identified by [BUR03] that guide the development of Model Driven Access Control:

1. “Access Control Points should be identifiable via parameterization of the domain PIMs with a single model element.”
2. “Protected Resources should be identifiable via parameterization of the domain PIMs with a single model element.”
3. “Access policy should be defined, developed and managed separately from the application business logic. Access policy rules and access policy models must be able to evolve without modification of the business software.”
4. “The policy model (role-based, user-based, code-based…) should not be exposed to the business application developers. That is, the business logic should have

absolutely no knowledge of the access control model utilized to make access decisions."

5. "The access control platform independent model must provide abstractions that are consistent with existing commonly used platform specific models for access control."

Thinking of the Prolog and TLA models as platform independent models, access control points are identified through the use of the Guard operators in both models. Also, resources are identified through simple constructs in each modeling language. In addition, this thesis keeps the access control policy separated from component models and implementations. The policy model is unknown to components because security attributes for each principal are provided by each principal at runtime. These attributes are simply submitted to the access control points and used during policy evaluation, which happens outside of the component, so the business logic need not be concerned with these attributes. A component model may also have a *DynamicContextServer* for adding dynamic attributes to the security attribute list for a principal when needed. Through applying these principles, this thesis successfully developed and tested an access control model of UniFrame systems indicating that the principles of model driven access control are potentially flexible enough to be used in a variety of contexts.

With the guidance of these principles, the access control framework proposed in this thesis is a promising approach for making access control characteristics of individual software components composable, thereby making the properties of systems specifiable and predictable. The student information system case study successfully demonstrated the effectiveness of this framework.

LIST OF REFERENCES

[ABA93] Abadi, A., Lampson, B., Plotkin G. “A Calculus for Access Control in Distributed Systems,” ACM Transactions: Programming Languages and Systems, vol. 15, no. 4, pp 706-734, 1994.

[ABE03] Abendroth, J., Jensen, C. “A Unified Security Framework for Networked Applications,” Proceedings of the 2003 ACM symposium on Applied Computing, pp 351-357, Melbourne, Florida, 2003.

[ANS03] American National Standards Institute (ANSI), “Role Based Access Control,” Draft Technical Report, April 2003.

[BON00] Bonatti, P., Vimercati, S., Samarati, P. “A Modular Approach to Composing Access Control Policies,” Proceedings of the 7th ACM conference on Computer and communications security, Athens, Greece, November 1-4, 2000.

[BRA01] Brahnmath, G. “The UniFrame Quality Of Service Framework.” M. S. Thesis, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, December 2002.

[BUR03] Burt, C., Bryant, B., Raje, R., Olson, A., and Auguston, M. “Model Driven Security: Unification of Authorization Models for Fine-Grain Access Control,” Proceedings of EDOC 2003, The 7th IEEE International Enterprise Distributed Object Computing Conference, Brisbane, Australia, September 16-19, 2003.

[CC04] Common Criteria Project Sponsoring Organizations, "Common Criteria for Information Technology Security Evaluation, Part 2: Security Functional Requirements," Version 2.2, January 2004.

[HAR03] Harrington, A., Jensen C. "Cryptographic Access Control in a Distributed File System," Proceedings of the eighth ACM symposium on Access control models and technologies, pp 158-165, Como, Italy, 2003.

[HUA03] Huang, Z., "The UniFrame System Generative Programming Framework," M. S. Thesis, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, April 2003.

[IET99] (IETF), "A URN Namespace for IETF Documents," Request for Comment 2648, August 2001, http://www.ietf.org/rfc/rfc2648.txt.

[IET01a] (IETF), "The Network Solutions Personal Internet Name (PIN): A URN Namespace for People and Organizations," Request for Comment 3043, January 2001, http://www.ietf.org/rfc/rfc3043.txt.

[IET01b] (IETF), "A URN Namespace of Object Identifiers," Request for Comment 3061, February 2001, http://www.ietf.org/rfc/rfc3061.txt.

[IET01c] (IETF), "A URN Namespace for OASIS," Request for Comment 3121, June 2001, http://www.ietf.org/rfc/rfc3121.txt.

[KAT02] Katzke, S. "Future Directions of the Common Criteria (CC) and the Common Evaluation Methodology (CEM)," National Institute of Standards and Technology, 2002.

[KHA02] Khan, K., Han, J. "Composing Security-Aware Software," IEEE Software, vol. 19, no. 1, pp 34-41, 2002.

[KHA03] Khan, K., Han, J. "Security Characterisation Framework for Trustworthy Component Based Software System," Technical Report No. CIT/35/2003, University of Western Sydney, School of Computing and Information Technology, 2003.

[KRI98] Kristensen, L., Christensen, S., Jensen, K.,"The practitioner's guide to coloured Petri nets," International Journal on Software Tools for Technology Transfer, no. 2, pp 98-132, 1998.

[LAM94] Lamport, L. "The Temporal Logic of Actions," ACM Transactions on Programming Languages and Systems, vol. 16, no. 3, pp 872-923, 1994.

[LAM03] Lamport, L. "Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers," Addison-Wesley, 2003.

[MIN98] Minsky, N., Ungureanu, V. "Unified Support for Heterogeneous Security Policies in Distributed Systems," Proceedings of the 7th USENIX Security Symposium, Berkeley, California, 1998.

[MUR89] Murata, T. "Petri Nets: Properties, Analysis and Applications," Proceedings of the IEEE, vol. 77, no. 4, pp 541-580, 1989.

[OMG01a] Object Management Group (OMG), "Model-Driven Architecture: A Technical Perspective," Technical Report, OMG Document No. ab/2001-02-01/04, February 2001, ftp://ftp.omg.org/pub/docs/ab/01-02-04.pdf.

[OMG01b] Object Management Group (OMG), "Resource Access Decision Facility Specification," Technical Report, 2001, http://www.omg.org/docs/formal/01-04-01.pdf.

[PET62] Petri, C. "Communications with Automata," Technical Report RADC-TR-65-377, Applied Data Research, Princeton, NJ, 1962.

[RAJ00] Raje, R. “UMM: Unified Meta-Object Model for Open Distributed Systems,” Proceedings of 4th IEEE International Conference on Algorithms and Architecture for Parallel Processing, ICA3PP 2000, pp 454-465, Hong Kong, 2000.

[RAJ01] Raje, R., Auguston, M., Bryant B., Olson, A., Burt, C. “A Unified Approach for the Integration of Distributed Heterogeneous Software Components,” Proceedings of the 2001 Monterey Workshop Engineering Automation for Software Intensive System Integration, pp 109-119, Monterey, California, 2001.

[RAJ02] Raje, R., Bryant, B., Olson, A., Auguston, M., Burt, C. “A Quality of Service-based Framework for Creating Distributed Heterogeneous Software Components,” Concurrency and Computation: Practice and Experience, vol. 14, pp 1009-1034, 2002.

[SAN94] Sandhu, R., Samarati, P. “Access Control: Principles and Practice,” IEEE Communications Magazine, vol. 32, no. 9, pp 40-48, 1994.

[SAN96] Sandhu, R., Coyne E., Feinstein, H., Youman, C. “Role-Based Access Control Models,” IEEE Computer, vol. 29, no. 2, pp 38-47, 1996.

[SAU01] Saunders, G., Hitchens, M., Varadharajan, V. “Role-based access control and the Access Control Matrix,” ACM SIGOPS Operating Systems Review, vol. 35, no. 4, pp 6-20, 2001.

[SEB02] Robert, S. “Concepts of Programming Languages,” Addison Wesley, 2002.

[SIR02] Siram, N. “An Architecture for Discovery of Heterogeneous Software Components,” M. S. Thesis, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, March, 2002.

[SUN03] Sun, C. “QOS Composition and Decomposition in UniFrame,” M. S. Thesis, Department of Computer & Information Science, Indiana University Purdue University Indianapolis, June, 2003.

[TAY03] Taylor, K., Murty J. “Implementing Role Based Access Control for Federated Information Systems on the Web,” Conference in Research and Practice in Information Technology Series, Proceedings of the Australasian information security workshop conference on ACSW frontiers 2003, vol. 21, pp 87-95, Adelaide, Australia, 2003.

[VIM97] Vimercati, S., Samarati, P. “Access Control in Federated Systems,” Proceedings of the 1996 workshop on New security paradigms, pp 87-99, Lake Arrowhead, California, 1997.

[W3C01] URI Planning Interest Group, W3C/IETF, “URIs, URLs, and URNs: Clarifications and Recommendations 1.0,” W3C Notes, September 2001.

Appendix A: Component UMM Specifications in XML format

One full specification is provided for each component type. In the cases when more than one component of that type was developed, only the modified specifications are given. Assume all other parts of the specification remain unchanged.

UserTerminal1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for User Terminal in the university domain example -->

<UMM_ConcreteComponent>
   <ComponentName> UserTerminal </ComponentName>
   <ComponentSubcase> UserTerminalCase1 </ComponentSubcase>
   <DomainName> University </DomainName>
   <SystemName> StudentInformation </SystemName>
   <Description> Provide GUI for users of the student information system. </Description>
   <ComputationalAttributes>
      <InherentAttributes>
         <id> UserTerminal1 </id>
         <Version> 1.0 </Version>
         <Author> Alex Crespi </Author>
         <Date> October 2004 </Date>
         <Validity> </Validity>
         <Atomicity> Yes </Atomicity>
         <Registration> localhost/HeadHunter1 </Registration>
         <Model> Java RMI </Model>
      </InherentAttributes>
      <FunctionalAttributes>
         <Purpose> Act as GUI terminal for student information system users. </Purpose>
         <Algorithms>
            <algorithm> JFC </algorithm>
         </Algorithms>
         <Complexity> O(1) </Complexity>
         <SyntacticContract>
            <ProvidedInterfaces>
               <Interface> IStudentManagement </Interface>
               <Interface> IValidation </Interface>
               <Interface> IStudentRecord </Interface>
               <Interface> IGeneralStudentInformationManagement </Interface>
               <Interface> IFinancialAidManagement </Interface>
               <Interface> IStudentEmploymentManagement </Interface>
               <Interface> IAcademicRecordManagement </Interface>
            </ProvidedInterfaces>
            <RequiredInterfaces>
               <Interface> IStudentManagement </Interface>
               <Interface> IValidation </Interface>
               <Interface> IStudentRecord </Interface>
               <Interface> IGeneralStudentInformationManagement </Interface>
```

```
                <Interface> IFinancialAidManagement </Interface>
                <Interface> IStudentEmploymentManagement </Interface>
                <Interface> IAcademicRecordManagement </Interface>
            </RequiredInterfaces>
        </SyntacticContract>
        <Technologies>
            <technology> Java RMI </technology>
        </Technologies>
        <ExpectedResources>
            <resource> CPU: 500mhz </resource>
            <resource> Memory: 1Gb </resource>
        </ExpectedResources>
        <DesignPatterns>
            <pattern> </pattern>
        </DesignPatterns>
        <KnownUsage>
            <usage> </usage>
        </KnownUsage>
        <Aliases>
            <alias> </alias>
        </Aliases>
     </FunctionalAttributes>
</ComputationalAttributes>
<CooperationAttributes>
    <PreprocessingCollaborators>
        <Collaborator> </Collaborator>
    </PreprocessingCollaborators>
    <PostprocessingCollaborators>
        <Collaborator> RecordServer </Collaborator>
        <Collaborator> ValidationServer </Collaborator>
    </PostprocessingCollaborators>
</CooperationAttributes>
<AuxiliaryAttributes>
    <Mobility> No </Mobility>
    <Security>
        <AccessControl>
            <ContainedResources>
            </ContainedResources>
            <ProtectedResources>
            </ProtectedResources>
        </AccessControl>
    </Security>
    <FaultTolerance> L1 </FaultTolerance>
</AuxiliaryAttributes>
<QoS>
    <QoSMetrics>
        <Metric>
            <ParameterName> throughput </ParameterName>
            <FunctionName> loadRecord </FunctionName>
            <Value> 4728.82 </Value>
        </Metric>
        <Metric>
            <ParameterName> throughput </ParameterName>
            <FunctionName> saveRecord </FunctionName>
            <Value> 3303.34 </Value>
         </Metric>
         <Metric>
            <ParameterName> endToEndDelay </ParameterName>
            <FunctionName> loadRecord </FunctionName>
            <Value> 211.47 </Value>
         </Metric>
         <Metric>
            <ParameterName> endToEndDelay </ParameterName>
            <FunctionName> saveRecord </FunctionName>
            <Value> 302.72 </Value>
         </Metric>
      </QoSMetrics>
      <QoSLevel> </QoSLevel>
      <Cost> </Cost>
      <QualityLevel> </QualityLevel>
   </QoS>
</UMM_ConcreteComponent>
```

RecordServer1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for Record Server in the university domain example -->

<UMM_ConcreteComponent>
   <ComponentName> RecordServer </ComponentName>
   <ComponentSubcase> RecordServerCase1 </ComponentSubcase>
   <DomainName> University </DomainName>
   <SystemName> StudentInformation </SystemName>
   <Description> provides storage of gsi for the student information system </Description>
   <ComputationalAttributes>
      <InherentAttributes>
         <id> RecordServer1 </id>
         <Version> 1.0 </Version>
         <Author> Alex Crespi </Author>
         <Date> October 2004 </Date>
         <Validity> </Validity>
         <Atomicity> Yes </Atomicity>
         <Registration> localhost/HeadHunter1 </Registration>
         <Model> Java RMI </Model>
      </InherentAttributes>
      <FunctionalAttributes>
         <Purpose> provides storage of gsi for the student information system</Purpose>
         <Algorithms>
            <algorithm> JFC </algorithm>
         </Algorithms>
         <Complexity> O(1) </Complexity>
         <SyntacticContract>
            <ProvidedInterfaces>
               <Interface> IStudentManagement </Interface>
               <Interface> IStudentRecord </Interface>
            <Interface> IGeneralStudentInformationManagement </Interface>
            <Interface> IFinancialAidManagement </Interface>
            <Interface> IStudentEmploymentManagement </Interface>
            <Interface> IAcademicRecordManagement </Interface>
         </ProvidedInterfaces>
         <RequiredInterfaces>
            <Interface> IStudentManagement </Interface>
            <Interface> IStudentRecord </Interface>
            <Interface> IGeneralStudentInformationManagement </Interface>
            <Interface> IFinancialAidManagement </Interface>
            <Interface> IStudentEmploymentManagement </Interface>
            <Interface> IAcademicRecordManagement </Interface>
         </RequiredInterfaces>
      </SyntacticContract>
      <Technologies>
         <technology> Java RMI </technology>
      </Technologies>
      <ExpectedResources>
         <resource> CPU: 500mhz </resource>
         <resource> Memory: 1Gb </resource>
      </ExpectedResources>
      <DesignPatterns>
         <pattern> </pattern>
      </DesignPatterns>
      <KnownUsage>
         <usage> </usage>
      </KnownUsage>
      <Aliases>
         <alias> </alias>
      </Aliases>
     </FunctionalAttributes>
</ComputationalAttributes>
<CooperationAttributes>
   <PreprocessingCollaborators>
      <Collaborator> UserTerminal </Collaborator>
   </PreprocessingCollaborators>
   <PostprocessingCollaborators>
      <Collaborator> AcademicRecordServer </Collaborator>
      <Collaborator> FinancialAidServer </Collaborator>
      <Collaborator> StudentEmploymentServer </Collaborator>
   </PostprocessingCollaborators>
</CooperationAttributes>
<AuxiliaryAttributes>
   <Mobility> No </Mobility>
   <Security>
      <AccessControl>
         <ContainedResources>
            <Resource>gsi</Resource>
         </ContainedResources>
         <ProtectedResources>
         </ProtectedResources>
      </AccessControl>
   </Security>
   <FaultTolerance> L1 </FaultTolerance>
```

```
  </AuxiliaryAttributes>
  <QoS>
    <QoSMetrics>
      <Metric>
        <ParameterName> throughput </ParameterName>
        <FunctionName> loadRecord </FunctionName>
        <Value> 4728.82 </Value>
      </Metric>
      <Metric>
        <ParameterName> throughput </ParameterName>
        <FunctionName> saveRecord </FunctionName>
        <Value> 3303.34 </Value>
      </Metric>
      <Metric>
        <ParameterName> endToEndDelay </ParameterName>
        <FunctionName> loadRecord </FunctionName>
        <Value> 211.47 </Value>
      </Metric>
      <Metric>
        <ParameterName> endToEndDelay </ParameterName>
        <FunctionName> saveRecord </FunctionName>
        <Value> 302.72 </Value>
      </Metric>
    </QoSMetrics>
    <QoSLevel> </QoSLevel>
    <Cost> </Cost>
    <QualityLevel> </QualityLevel>
  </QoS>
</UMM_ConcreteComponent>
```

RecordServer2

```
    <Security>
      <AccessControl>
        <ContainedResources>
          <Resource>gsi</Resource>
        </ContainedResources>
        <ProtectedResources>
          <Resource>gsi</Resource>
        </ProtectedResources>
      </AccessControl>
    </Security>
```

RecordServer3

```
    <Security>
      <AccessControl>
        <ContainedResources>
          <Resource>gsi</Resource>
          <Resource>ar</Resource>
          <Resource>fa</Resource>
```

```
            <Resource>se</Resource>
         </ContainedResources>
         <ProtectedResources>
            <Resource>gsi</Resource>
         </ProtectedResources>
      </AccessControl>
   </Security>
```

RecordServer4

```
   <Security>
      <AccessControl>
         <ContainedResources>
            <Resource>gsi</Resource>
            <Resource>ar</Resource>
            <Resource>fa</Resource>
            <Resource>se</Resource>
         </ContainedResources>
         <ProtectedResources>
            <Resource>gsi</Resource>
            <Resource>ar</Resource>
            <Resource>fa</Resource>
            <Resource>se</Resource>
         </ProtectedResources>
      </AccessControl>
   </Security>
```

AcademicRecordServer1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for Academic Record Server in the university domain example -->

<UMM_ConcreteComponent>
   <ComponentName> AcademicRecordServer </ComponentName>
   <ComponentSubcase> AcademicRecordServerCase1 </ComponentSubcase>
   <DomainName> University </DomainName>
   <SystemName> StudentInformation </SystemName>
   <Description> Provide storage of ar for the student information system. </Description>
   <ComputationalAttributes>
      <InherentAttributes>
         <id> AcademicRecordServer1 </id>
         <Version> 1.0 </Version>
         <Author> Alex Crespi </Author>
         <Date> October 2004 </Date>
         <Validity> </Validity>
         <Atomicity> Yes </Atomicity>
         <Registration> localhost/HeadHunter1 </Registration>
         <Model> Java RMI </Model>
      </InherentAttributes>
```

```
    <FunctionalAttributes>
      <Purpose> Provide storage of ar for the student information system. </Purpose>
      <Algorithms>
        <algorithm> JFC </algorithm>
      </Algorithms>
      <Complexity> O(1) </Complexity>
      <SyntacticContract>
        <ProvidedInterfaces>
          <Interface> IStudentManagement </Interface>
          <Interface> IAcademicRecordManagement </Interface>
        </ProvidedInterfaces>
        <RequiredInterfaces>
        </RequiredInterfaces>
      </SyntacticContract>
      <Technologies>
        <technology> Java RMI </technology>
      </Technologies>
      <ExpectedResources>
        <resource> CPU: 500mhz </resource>
        <resource> Memory: 1Gb </resource>
      </ExpectedResources>
      <DesignPatterns>
        <pattern> </pattern>
      </DesignPatterns>
      <KnownUsage>
        <usage> </usage>
      </KnownUsage>
      <Aliases>
        <alias> </alias>
      </Aliases>
     </FunctionalAttributes>
  </ComputationalAttributes>
  <CooperationAttributes>
    <PreprocessingCollaborators>
      <Collaborator> RecordServer </Collaborator>
    </PreprocessingCollaborators>
    <PostprocessingCollaborators>
    </PostprocessingCollaborators>
  </CooperationAttributes>
  <AuxiliaryAttributes>
    <Mobility> No </Mobility>
    <Security>
      <AccessControl>
        <ContainedResources>
          <Resource>ar</Resource>
        </ContainedResources>
        <ProtectedResources>
        </ProtectedResources>
      </AccessControl>
    </Security>
    <FaultTolerance> L1 </FaultTolerance>
  </AuxiliaryAttributes>
  <QoS>
    <QoSMetrics>
```

```
        <Metric>
            <ParameterName> throughput </ParameterName>
            <FunctionName> loadRecord </FunctionName>
            <Value> 4728.82 </Value>
        </Metric>
        <Metric>
            <ParameterName> throughput </ParameterName>
            <FunctionName> saveRecord </FunctionName>
            <Value> 3303.34 </Value>
        </Metric>
        <Metric>
            <ParameterName> endToEndDelay </ParameterName>
            <FunctionName> loadRecord </FunctionName>
            <Value> 211.47 </Value>
        </Metric>
        <Metric>
            <ParameterName> endToEndDelay </ParameterName>
            <FunctionName> saveRecord </FunctionName>
            <Value> 302.72 </Value>
        </Metric>
    </QoSMetrics>
    <QoSLevel> </QoSLevel>
    <Cost> </Cost>
    <QualityLevel> </QualityLevel>
  </QoS>
</UMM_ConcreteComponent>
```

AcademicRecordServer2

```
    <Security>
        <AccessControl>
            <ContainedResources>
                <Resource>ar</Resource>
            </ContainedResources>
            <ProtectedResources>
                <Resource>ar</Resource>
            </ProtectedResources>
        </AccessControl>
    </Security>
```

FinancialAidServer1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for Financial Aid Server in the university domain example -->

<UMM_ConcreteComponent>
    <ComponentName> FinancialAidServer </ComponentName>
    <ComponentSubcase> FinancialAidServerCase1 </ComponentSubcase>
    <DomainName> University </DomainName>
    <SystemName> StudentInformation </SystemName>
```

```
<Description> Provide store of financial aid info the student information system. </Description>
<ComputationalAttributes>
   <InherentAttributes>
      <id> FinancialAidServer1 </id>
      <Version> 1.0 </Version>
      <Author> Alex Crespi </Author>
      <Date> October 2004 </Date>
      <Validity> </Validity>
      <Atomicity> Yes </Atomicity>
      <Registration> localhost/HeadHunter1 </Registration>
      <Model> Java RMI </Model>
   </InherentAttributes>
   <FunctionalAttributes>
      <Purpose> Provide store of financial aid info the student information system. </Purpose>
      <Algorithms>
         <algorithm> JFC </algorithm>
      </Algorithms>
      <Complexity> O(1) </Complexity>
      <SyntacticContract>
         <ProvidedInterfaces>
            <Interface> IStudentManagement </Interface>
            <Interface> IFinancialAidManagement </Interface>
         </ProvidedInterfaces>
         <RequiredInterfaces>
         </RequiredInterfaces>
      </SyntacticContract>
      <Technologies>
         <technology> Java RMI </technology>
      </Technologies>
      <ExpectedResources>
         <resource> CPU: 500mhz </resource>
         <resource> Memory: 1Gb </resource>
      </ExpectedResources>
      <DesignPatterns>
         <pattern> </pattern>
      </DesignPatterns>
      <KnownUsage>
         <usage> </usage>
      </KnownUsage>
      <Aliases>
         <alias> </alias>
      </Aliases>
    </FunctionalAttributes>
</ComputationalAttributes>
<CooperationAttributes>
   <PreprocessingCollaborators>
      <Collaborator> RecordServer </Collaborator>
   </PreprocessingCollaborators>
   <PostprocessingCollaborators>
   </PostprocessingCollaborators>
</CooperationAttributes>
<AuxiliaryAttributes>
   <Mobility> No </Mobility>
   <Security>
```

```
            <AccessControl>
                <ContainedResources>
                    <Resource>fa</Resource>
                </ContainedResources>
                <ProtectedResources>
                </ProtectedResources>
            </AccessControl>
        </Security>
        <FaultTolerance> L1 </FaultTolerance>
    </AuxiliaryAttributes>
    <QoS>
        <QoSMetrics>
            <Metric>
                <ParameterName> throughput </ParameterName>
                <FunctionName> loadRecord </FunctionName>
                <Value> 4728.82 </Value>
            </Metric>
            <Metric>
                <ParameterName> throughput </ParameterName>
                <FunctionName> saveRecord </FunctionName>
                <Value> 3303.34 </Value>
            </Metric>
            <Metric>
                <ParameterName> endToEndDelay </ParameterName>
                <FunctionName> loadRecord </FunctionName>
                <Value> 211.47 </Value>
            </Metric>
            <Metric>
                <ParameterName> endToEndDelay </ParameterName>
                <FunctionName> saveRecord </FunctionName>
                <Value> 302.72 </Value>
            </Metric>
        </QoSMetrics>
        <QoSLevel> </QoSLevel>
        <Cost> </Cost>
        <QualityLevel> </QualityLevel>
    </QoS>
</UMM_ConcreteComponent>
```

FinancialAidServer2

```
        <Security>
            <AccessControl>
                <ContainedResources>
                    <Resource>fa</Resource>
                </ContainedResources>
                <ProtectedResources>
                    <Resource>fa</Resource>
                </ProtectedResources>
            </AccessControl>
        </Security>
```

StudentEmploymentServer1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for Student Employment Server in the university domain example -->

<UMM_ConcreteComponent>
  <ComponentName> StudentEmploymentServer </ComponentName>
  <ComponentSubcase> StudentEmploymentAidServerCase1 </ComponentSubcase>
  <DomainName> University </DomainName>
  <SystemName> StudentInformation </SystemName>
  <Description> Provide storage for the student employment info the student information system.
</Description>
  <ComputationalAttributes>
    <InherentAttributes>
      <id> StudentEmploymentServer1 </id>
      <Version> 1.0 </Version>
      <Author> Alex Crespi </Author>
      <Date> October 2004 </Date>
      <Validity> </Validity>
      <Atomicity> Yes </Atomicity>
      <Registration> localhost/HeadHunter1 </Registration>
      <Model> Java RMI </Model>
    </InherentAttributes>
    <FunctionalAttributes>
      <Purpose> Provide storage for the student employment info the student information system.
</Purpose>
      <Algorithms>
        <algorithm> JFC </algorithm>
      </Algorithms>
      <Complexity> O(1) </Complexity>
      <SyntacticContract>
        <ProvidedInterfaces>
          <Interface> IStudentManagement </Interface>
          <Interface> IStudentEmploymentManagement </Interface>
        </ProvidedInterfaces>
        <RequiredInterfaces>
        </RequiredInterfaces>
      </SyntacticContract>
      <Technologies>
        <technology> Java RMI </technology>
      </Technologies>
      <ExpectedResources>
        <resource> CPU: 500mhz </resource>
        <resource> Memory: 1Gb </resource>
      </ExpectedResources>
      <DesignPatterns>
        <pattern> </pattern>
      </DesignPatterns>
      <KnownUsage>
        <usage> </usage>
      </KnownUsage>
      <Aliases>
```

```
            <alias> </alias>
          </Aliases>
        </FunctionalAttributes>
    </ComputationalAttributes>
    <CooperationAttributes>
      <PreprocessingCollaborators>
        <Collaborator> RecordServer </Collaborator>
      </PreprocessingCollaborators>
      <PostprocessingCollaborators>
      </PostprocessingCollaborators>
    </CooperationAttributes>
    <AuxiliaryAttributes>
      <Mobility> No </Mobility>
      <Security>
        <AccessControl>
          <ContainedResources>
            <Resource>se</Resource>
          </ContainedResources>
            <Resource>se</Resource>
          <ProtectedResources>
          </ProtectedResources>
        </AccessControl>
      </Security>
      <FaultTolerance> L1 </FaultTolerance>
    </AuxiliaryAttributes>
    <QoS>
      <QoSMetrics>
        <Metric>
          <ParameterName> throughput </ParameterName>
          <FunctionName> loadRecord </FunctionName>
          <Value> 4728.82 </Value>
        </Metric>
        <Metric>
          <ParameterName> throughput </ParameterName>
          <FunctionName> saveRecord </FunctionName>
          <Value> 3303.34 </Value>
        </Metric>
        <Metric>
          <ParameterName> endToEndDelay </ParameterName>
          <FunctionName> loadRecord </FunctionName>
          <Value> 211.47 </Value>
        </Metric>
        <Metric>
          <ParameterName> endToEndDelay </ParameterName>
          <FunctionName> saveRecord </FunctionName>
          <Value> 302.72 </Value>
        </Metric>
      </QoSMetrics>
      <QoSLevel> </QoSLevel>
      <Cost> </Cost>
      <QualityLevel> </QualityLevel>
    </QoS>
  </UMM_ConcreteComponent>
```

StudentEmploymentServer2

```
<Security>
   <AccessControl>
      <ContainedResources>
         <Resource>se</Resource>
      </ContainedResources>
      <ProtectedResources>
         <Resource>se</Resource>
      </ProtectedResources>
   </AccessControl>
</Security>
```

Appendix B: Component Queries in XMl Format

One full query specification is provided for each component type. In the cases when more than one query was created for a particular component type, only the modified specifications are given. Assume all other parts of the specification remain unchanged.

UserTerminal - Query 1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for User Terminal in the university domain example -->

<UMM_AbstractComponent>
  <ComponentName> UserTerminal </ComponentName>
  <ComponentSubcase> UserTerminalCase1 </ComponentSubcase>
  <DomainName> University </DomainName>
  <SystemName> StudentInformation </SystemName>
  <Description> Provide GUI for users of the student information system. </Description>
  <ComputationalAttributes>
    <InherentAttributes>
      <id> </id>
      <Version> </Version>
      <Author> </Author>
      <Date> </Date>
      <Validity> </Validity>
      <Atomicity> </Atomicity>
      <Registration> </Registration>
      <Model> </Model>
    </InherentAttributes>
    <FunctionalAttributes>
      <Purpose> Act as GUI terminal for student information system users. </Purpose>
      <Algorithms>
        <algorithm> </algorithm>
      </Algorithms>
      <Complexity> O(1) </Complexity>
      <SyntacticContract>
        <ProvidedInterfaces>
          <Interface> IStudentManagement </Interface>
          <Interface> IValidation </Interface>
          <Interface> IStudentRecord </Interface>
          <Interface> IGeneralStudentInformationManagement </Interface>
          <Interface> IFinancialAidManagement </Interface>
          <Interface> IStudentEmploymentManagement </Interface>
          <Interface> IAcademicRecordManagement </Interface>
        </ProvidedInterfaces>
        <RequiredInterfaces>
          <Interface> IStudentManagement </Interface>
          <Interface> IValidation </Interface>
          <Interface> IStudentRecord </Interface>
```

```
                <Interface> IGeneralStudentInformationManagement </Interface>
                <Interface> IFinancialAidManagement </Interface>
                <Interface> IStudentEmploymentManagement </Interface>
                <Interface> IAcademicRecordManagement </Interface>
            </RequiredInterfaces>
        </SyntacticContract>
        <Technologies>
            <technology> Java RMI </technology>
        </Technologies>
        <ExpectedResources>
            <resource> CPU: 500mhz </resource>
            <resource> Memory: 1Gb </resource>
        </ExpectedResources>
        <DesignPatterns>
            <pattern> </pattern>
        </DesignPatterns>
        <KnownUsage>
            <usage> </usage>
        </KnownUsage>
        <Aliases>
            <alias> </alias>
        </Aliases>
     </FunctionalAttributes>
</ComputationalAttributes>
<CooperationAttributes>
    <PreprocessingCollaborators>
        <Collaborator> </Collaborator>
    </PreprocessingCollaborators>
    <PostprocessingCollaborators>
        <Collaborator> RecordServer </Collaborator>
        <Collaborator> ValidationServer </Collaborator>
    </PostprocessingCollaborators>
</CooperationAttributes>
<AuxiliaryAttributes>
    <Mobility> No </Mobility>
    <Security>
        <AccessControl>
            <ContainedResources>
            </ContainedResources>
            <ProtectedResources>
            </ProtectedResources>
        </AccessControl>
    </Security>
    <FaultTolerance> L1 </FaultTolerance>
</AuxiliaryAttributes>
<QoS>
    <QoSMetrics>
        <Metric>
            <ParameterName> throughput </ParameterName>
            <FunctionName> loadRecord </FunctionName>
            <Value> 4728.82 </Value>
        </Metric>
        <Metric>
            <ParameterName> throughput </ParameterName>
            <FunctionName> saveRecord </FunctionName>
            <Value> 3303.34 </Value>
         </Metric>
         <Metric>
            <ParameterName> endToEndDelay </ParameterName>
            <FunctionName> loadRecord </FunctionName>
            <Value> 211.47 </Value>
         </Metric>
         <Metric>
            <ParameterName> endToEndDelay </ParameterName>
            <FunctionName> saveRecord </FunctionName>
            <Value> 302.72 </Value>
         </Metric>
      </QoSMetrics>
      <QoSLevel> </QoSLevel>
      <Cost> </Cost>
      <QualityLevel> </QualityLevel>
   </QoS>
</UMM_AbstractComponent>
```

RecordServer - Query 1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for Record Server in the university domain example -->

<UMM_AbstractComponent>
   <ComponentName> RecordServer </ComponentName>
   <ComponentSubcase> RecordServerCase1 </ComponentSubcase>
   <DomainName> University </DomainName>
   <SystemName> StudentInformation </SystemName>
   <Description> Provide protection of gsi instudent information system. </Description>
   <ComputationalAttributes>
      <InherentAttributes>
         <id> </id>
         <Version> </Version>
         <Author> </Author>
         <Date> </Date>
         <Validity> </Validity>
         <Atomicity> </Atomicity>
         <Registration> </Registration>
         <Model> </Model>
      </InherentAttributes>
      <FunctionalAttributes>
         <Purpose> Provide protection of gsi instudent information system. </Purpose>
         <Algorithms>
            <algorithm> </algorithm>
         </Algorithms>
         <Complexity> O(1) </Complexity>
         <SyntacticContract>
            <ProvidedInterfaces>
               <Interface> IStudentManagement </Interface>
```

```
        <Interface> IStudentRecord </Interface>
        <Interface> IGeneralStudentInformationManagement </Interface>
        <Interface> IFinancialAidManagement </Interface>
        <Interface> IStudentEmploymentManagement </Interface>
        <Interface> IAcademicRecordManagement </Interface>
      </ProvidedInterfaces>
      <RequiredInterfaces>
        <Interface> IStudentManagement </Interface>
        <Interface> IStudentRecord </Interface>
        <Interface> IGeneralStudentInformationManagement </Interface>
        <Interface> IFinancialAidManagement </Interface>
        <Interface> IStudentEmploymentManagement </Interface>
        <Interface> IAcademicRecordManagement </Interface>
      </RequiredInterfaces>
    </SyntacticContract>
    <Technologies>
      <technology> Java RMI </technology>
    </Technologies>
    <ExpectedResources>
      <resource> CPU: 500mhz </resource>
      <resource> Memory: 1Gb </resource>
    </ExpectedResources>
    <DesignPatterns>
      <pattern> </pattern>
    </DesignPatterns>
    <KnownUsage>
      <usage> </usage>
    </KnownUsage>
    <Aliases>
      <alias> </alias>
    </Aliases>
   </FunctionalAttributes>
</ComputationalAttributes>
<CooperationAttributes>
  <PreprocessingCollaborators>
    <Collaborator> UserTerminal </Collaborator>
  </PreprocessingCollaborators>
  <PostprocessingCollaborators>
    <Collaborator> AcademicRecordServer </Collaborator>
    <Collaborator> FinancialAidServer </Collaborator>
    <Collaborator> StudentEmploymentServer </Collaborator>
  </PostprocessingCollaborators>
</CooperationAttributes>
<AuxiliaryAttributes>
  <Mobility> No </Mobility>
  <Security>
    <AccessControl>
      <ContainedResources>
      </ContainedResources>
      <ProtectedResources>
        <Resource>gsi</Resource>
      </ProtectedResources>
    </AccessControl>
  </Security>
```

```
      <FaultTolerance> L1 </FaultTolerance>
    </AuxiliaryAttributes>
    <QoS>
      <QoSMetrics>
        <Metric>
          <ParameterName> throughput </ParameterName>
          <FunctionName> loadRecord </FunctionName>
          <Value> 4728.82 </Value>
        </Metric>
        <Metric>
          <ParameterName> throughput </ParameterName>
          <FunctionName> saveRecord </FunctionName>
          <Value> 3303.34 </Value>
        </Metric>
        <Metric>
          <ParameterName> endToEndDelay </ParameterName>
          <FunctionName> loadRecord </FunctionName>
          <Value> 211.47 </Value>
        </Metric>
        <Metric>
          <ParameterName> endToEndDelay </ParameterName>
          <FunctionName> saveRecord </FunctionName>
          <Value> 302.72 </Value>
        </Metric>
      </QoSMetrics>
      <QoSLevel> </QoSLevel>
      <Cost> </Cost>
      <QualityLevel> </QualityLevel>
    </QoS>
  </UMM_AbstractComponent>
```

RecordServer - Query 2

```
      <Security>
        <AccessControl>
          <ContainedResources>
          </ContainedResources>
          <ProtectedResources>
            <Resource>gsi</Resource>
            <Resource>ar</Resource>
            <Resource>fa</Resource>
            <Resource>se</Resource>
          </ProtectedResources>
        </AccessControl>
      </Security>
```

RecordServer - Query 3

```
<Security>
  <AccessControl>
    <ContainedResources>
    </ContainedResources>
    <ProtectedResources>
      <Resource>gsi</Resource>
      <Resource>ar</Resource>
      <Resource>fa</Resource>
    </ProtectedResources>
  </AccessControl>
</Security>
```

AcademicRecordServer- Query 1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for Academic Record Server in the university domain example -->

<UMM_AbstractComponent>
  <ComponentName> AcademicRecordServer </ComponentName>
  <ComponentSubcase> AcademicRecordServerCase1 </ComponentSubcase>
  <DomainName> University </DomainName>
  <SystemName> StudentInformation </SystemName>
  <Description> </Description>
  <ComputationalAttributes>
    <InherentAttributes>
      <id>  </id>
      <Version> </Version>
      <Author> </Author>
      <Date> </Date>
      <Validity> </Validity>
      <Atomicity> </Atomicity>
      <Registration> </Registration>
      <Model>  </Model>
    </InherentAttributes>
    <FunctionalAttributes>
      <Purpose>  </Purpose>
      <Algorithms>
        <algorithm> </algorithm>
      </Algorithms>
      <Complexity> O(1) </Complexity>
      <SyntacticContract>
        <ProvidedInterfaces>
          <Interface> IStudentManagement </Interface>
          <Interface> IAcademicRecordManagement </Interface>
        </ProvidedInterfaces>
        <RequiredInterfaces>
        </RequiredInterfaces>
      </SyntacticContract>
      <Technologies>
```

```
        <technology> Java RMI </technology>
      </Technologies>
      <ExpectedResources>
        <resource> CPU: 500mhz </resource>
        <resource> Memory: 1Gb </resource>
      </ExpectedResources>
      <DesignPatterns>
        <pattern> </pattern>
      </DesignPatterns>
      <KnownUsage>
        <usage> </usage>
      </KnownUsage>
      <Aliases>
        <alias> </alias>
      </Aliases>
    </FunctionalAttributes>
</ComputationalAttributes>
<CooperationAttributes>
  <PreprocessingCollaborators>
    <Collaborator> RecordServer </Collaborator>
  </PreprocessingCollaborators>
  <PostprocessingCollaborators>
    <Collaborator> </Collaborator>
  </PostprocessingCollaborators>
</CooperationAttributes>
<AuxiliaryAttributes>
  <Mobility> No </Mobility>
  <Security>
    <AccessControl>
      <ContainedResources>
        <Resource>ar</Resource>
      </ContainedResources>
      <ProtectedResources>
      </ProtectedResources>
    </AccessControl>
  </Security>
  <FaultTolerance> L1 </FaultTolerance>
</AuxiliaryAttributes>
<QoS>
  <QoSMetrics>
    <Metric>
      <ParameterName> throughput </ParameterName>
      <FunctionName> loadRecord </FunctionName>
      <Value> 4728.82 </Value>
    </Metric>
    <Metric>
      <ParameterName> throughput </ParameterName>
      <FunctionName> saveRecord </FunctionName>
      <Value> 3303.34 </Value>
    </Metric>
    <Metric>
      <ParameterName> endToEndDelay </ParameterName>
      <FunctionName> loadRecord </FunctionName>
      <Value> 211.47 </Value>
```

```
            </Metric>
            <Metric>
                <ParameterName> endToEndDelay </ParameterName>
                <FunctionName> saveRecord </FunctionName>
                <Value> 302.72 </Value>
            </Metric>
        </QoSMetrics>
        <QoSLevel> </QoSLevel>
        <Cost> </Cost>
        <QualityLevel> </QualityLevel>
    </QoS>
</UMM_AbstractComponent>
```

AcademicRecordServer- Query 2

```
        <Security>
            <AccessControl>
                <ContainedResources>
                </ContainedResources>
                <ProtectedResources>
                    <Resource>gsi</Resource>
                    <Resource>ar</Resource>
                    <Resource>fa</Resource>
                    <Resource>se</Resource>
                </ProtectedResources>
            </AccessControl>
        </Security>
```

AcademicRecordServer- Query 3

```
        <Security>
            <AccessControl>
                <ContainedResources>
                </ContainedResources>
                <ProtectedResources>
                    <Resource>gsi</Resource>
                    <Resource>ar</Resource>
                    <Resource>fa</Resource>
                </ProtectedResources>
            </AccessControl>
        </Security>
```

FinancialAidServer- Query 1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for Financial Aid Server in the university domain example -->

<UMM_AbstractComponent>
  <ComponentName> FinancialAidServer </ComponentName>
  <ComponentSubcase> FinancialAidServerCase1 </ComponentSubcase>
  <DomainName> University </DomainName>
  <SystemName> StudentInformation </SystemName>
  <Description>  </Description>
  <ComputationalAttributes>
    <InherentAttributes>
      <id>   </id>
      <Version>   </Version>
      <Author>  </Author>
      <Date>  </Date>
      <Validity> </Validity>
      <Atomicity>   </Atomicity>
      <Registration>  </Registration>
      <Model>  </Model>
    </InherentAttributes>
    <FunctionalAttributes>
      <Purpose>  </Purpose>
      <Algorithms>
        <algorithm>   </algorithm>
      </Algorithms>
      <Complexity> O(1) </Complexity>
      <SyntacticContract>
        <ProvidedInterfaces>
          <Interface> IStudentManagement </Interface>
          <Interface> IFinancialAidManagement </Interface>
        </ProvidedInterfaces>
        <RequiredInterfaces>
        </RequiredInterfaces>
      </SyntacticContract>
      <Technologies>
        <technology> Java RMI </technology>
      </Technologies>
      <ExpectedResources>
        <resource> CPU: 500mhz </resource>
        <resource> Memory: 1Gb </resource>
      </ExpectedResources>
      <DesignPatterns>
        <pattern> </pattern>
      </DesignPatterns>
      <KnownUsage>
        <usage> </usage>
      </KnownUsage>
      <Aliases>
        <alias> </alias>
      </Aliases>
      </FunctionalAttributes>
   </ComputationalAttributes>
   <CooperationAttributes>
      <PreprocessingCollaborators>
         <Collaborator> RecordServer </Collaborator>
      </PreprocessingCollaborators>
      <PostprocessingCollaborators>
         <Collaborator> </Collaborator>
      </PostprocessingCollaborators>
   </CooperationAttributes>
   <AuxiliaryAttributes>
      <Mobility> No </Mobility>
      <Security>
         <AccessControl>
            <ContainedResources>
               <Resource>fa</Resource>
            </ContainedResources>
            <ProtectedResources>
            </ProtectedResources>
         </AccessControl>
      </Security>
      <FaultTolerance> L1 </FaultTolerance>
   </AuxiliaryAttributes>
   <QoS>
      <QoSMetrics>
         <Metric>
            <ParameterName> throughput </ParameterName>
            <FunctionName> loadRecord </FunctionName>
            <Value> 4728.82 </Value>
         </Metric>
         <Metric>
            <ParameterName> throughput </ParameterName>
            <FunctionName> saveRecord </FunctionName>
            <Value> 3303.34 </Value>
         </Metric>
         <Metric>
            <ParameterName> endToEndDelay </ParameterName>
            <FunctionName> loadRecord </FunctionName>
            <Value> 211.47 </Value>
         </Metric>
         <Metric>
            <ParameterName> endToEndDelay </ParameterName>
            <FunctionName> saveRecord </FunctionName>
            <Value> 302.72 </Value>
         </Metric>
      </QoSMetrics>
      <QoSLevel> </QoSLevel>
      <Cost> </Cost>
      <QualityLevel> </QualityLevel>
   </QoS>
</UMM_AbstractComponent>
```

FinancialAidServer - Query 2

```
<Security>
  <AccessControl>
    <ContainedResources>
    </ContainedResources>
    <ProtectedResources>
      <Resource>gsi</Resource>
      <Resource>ar</Resource>
      <Resource>fa</Resource>
      <Resource>se</Resource>
    </ProtectedResources>
  </AccessControl>
</Security>
```

FinancialAidServer - Query 3

```
<Security>
  <AccessControl>
    <ContainedResources>
    </ContainedResources>
    <ProtectedResources>
      <Resource>gsi</Resource>
      <Resource>ar</Resource>
      <Resource>fa</Resource>
    </ProtectedResources>
  </AccessControl>
</Security>
```

StudentEmploymentServer- Query 1

```
<?xml version="1.0" encoding='utf-8'?>

<!-- UMM description for Student Employment Server in the university domain example -->

<UMM_AbstractComponent>
  <ComponentName> StudentEmploymentServer </ComponentName>
  <ComponentSubcase> StudentEmploymentAidServerCase1 </ComponentSubcase>
  <DomainName> University </DomainName>
  <SystemName> StudentInformation </SystemName>
  <Description>  </Description>
  <ComputationalAttributes>
    <InherentAttributes>
      <id>   </id>
      <Version>  </Version>
      <Author>  </Author>
      <Date>  </Date>
      <Validity> </Validity>
      <Atomicity>   </Atomicity>
      <Registration>  </Registration>
```

```
            <Model>  </Model>
        </InherentAttributes>
        <FunctionalAttributes>
            <Purpose>  </Purpose>
            <Algorithms>
                <algorithm> JFC </algorithm>
            </Algorithms>
            <Complexity> O(1) </Complexity>
            <SyntacticContract>
                <ProvidedInterfaces>
                    <Interface> IStudentManagement </Interface>
                    <Interface> IStudentEmploymentManagement </Interface>
                </ProvidedInterfaces>
                <RequiredInterfaces>
                </RequiredInterfaces>
            </SyntacticContract>
            <Technologies>
                <technology> Java RMI </technology>
            </Technologies>
            <ExpectedResources>
                <resource> CPU: 500mhz </resource>
                <resource> Memory: 1Gb </resource>
            </ExpectedResources>
            <DesignPatterns>
                <pattern> </pattern>
            </DesignPatterns>
            <KnownUsage>
                <usage> </usage>
            </KnownUsage>
            <Aliases>
                <alias> </alias>
            </Aliases>
         </FunctionalAttributes>
    </ComputationalAttributes>
    <CooperationAttributes>
        <PreprocessingCollaborators>
            <Collaborator> RecordServer </Collaborator>
        </PreprocessingCollaborators>
        <PostprocessingCollaborators>
            <Collaborator> </Collaborator>
        </PostprocessingCollaborators>
    </CooperationAttributes>
    <AuxiliaryAttributes>
        <Mobility> No </Mobility>
        <Security>
            <AccessControl>
                <ContainedResources>
                    <Resource>se</Resource>
                </ContainedResources>
                <ProtectedResources>
                </ProtectedResources>
            </AccessControl>
        </Security>
        <FaultTolerance> L1 </FaultTolerance>
    </AuxiliaryAttributes>
    <QoS>
      <QoSMetrics>
        <Metric>
          <ParameterName> throughput </ParameterName>
          <FunctionName> loadRecord </FunctionName>
          <Value> 4728.82 </Value>
        </Metric>
        <Metric>
          <ParameterName> throughput </ParameterName>
          <FunctionName> saveRecord </FunctionName>
          <Value> 3303.34 </Value>
        </Metric>
        <Metric>
          <ParameterName> endToEndDelay </ParameterName>
          <FunctionName> loadRecord </FunctionName>
          <Value> 211.47 </Value>
        </Metric>
        <Metric>
          <ParameterName> endToEndDelay </ParameterName>
          <FunctionName> saveRecord </FunctionName>
          <Value> 302.72 </Value>
        </Metric>
      </QoSMetrics>
      <QoSLevel> </QoSLevel>
      <Cost> </Cost>
      <QualityLevel> </QualityLevel>
    </QoS>
  </UMM_AbstractComponent>
```

StudentEmploymentServer - Query 2

```
      <Security>
        <AccessControl>
          <ContainedResources>
          </ContainedResources>
          <ProtectedResources>
            <Resource>gsi</Resource>
            <Resource>ar</Resource>
            <Resource>fa</Resource>
            <Resource>se</Resource>
          </ProtectedResources>
        </AccessControl>
      </Security>
```

StudentEmploymentServer - Query 3

```
<Security>
  <AccessControl>
    <ContainedResources>
    </ContainedResources>
    <ProtectedResources>
      <Resource>gsi</Resource>
      <Resource>ar</Resource>
      <Resource>fa</Resource>
    </ProtectedResources>
  </AccessControl>
</Security>
```

Appendix C: Case Study Prolog Access Control Specifications

This appendix contains the Prolog access control model for test system 2. This is the complete specification composed from the components and used for testing in this thesis.

```
% FILE: studentRecordPrologACModel.pl
% DATE: Fri Nov 19 11:11:13 GMT-05:00 2004
% CREATED BY: UniFrame System Generator
% SYSTEM: Student Record System Access Control Model

%**********FACTS and PREDICATES for student records system**********
%*** These facts define system users and their security credentials ***
authenticate(acrespi,credential(accessID(acrespi),role([student]),group([student]))).
authenticate(hblake,credential(accessID(hblake),role([student]),group([student]))).
authenticate(fburns,credential(accessID(fburns),role([bursar]),group([staff]))).
authenticate(cwinchester,credential(accessID(cwinchester),role([academicadvisor]),group([staff]))).

%*** This predicate authenticates a principal in the system ***
principal(AID,P):- authenticate(AID,P).

%*** This predicate identifies students in the system ***
student(S):- authenticate(S,P),arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),member(student,ROLE).

%*** This predicate is used to represent allowed access to a resource ***
true(R,O):-write('ALLOWED:'),write(R),write(O),nl.

%*** This predicate is used to represent access control decisions ***
accessControl(P,R,O):-arg(1,P,AIDPRED),arg(1,AIDPRED,AID),
            arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLES),
            arg(3,P,GROUPPRED),arg(1,GROUPPRED,GROUP),
            nl,write('Security Attributes:'),
            nl,write('  ACCESSID='),write(AID),
            nl,write('  GROUP='),write(GROUP),
            nl,write('  ROLES='),write(ROLES),
            nl,write('  ResourceName='),write(R),

            %*** separate out the resource name components ***
            R=[SID,SECTION,PART|_],
            nl,nl,write('Resource:'),
            nl,write('  SR='),write(SID),
            nl,write('  Section='),write(SECTION),
            nl,write('  Part='),write(PART),nl,

            %*** Differentiate between students and staff ***
            (member(student,GROUP)-> checkstudent(ROLES,R,O);checkstaff(ROLES,R,O)).

%*** evalute policy for students trying to access the records ***

checkstudent(ROLES,R,O):- nl,write('checking student credentials for access. '),nl,
            R=[SID,SECTION,PART|_],
                write('  Resource='),write(R),nl,
                write('  ROLES='),write(ROLES),nl,
                write('  Operation='),write(O),nl,
                ( ( (member(studentowner,ROLES),(not SECTION==[section,notes]),O==read);
                   (member(studentowner,ROLES),SECTION==[section,gsi],O==save)       ) ->true(R,O)).


%*** evaluate policy for staff try to access the records ***

checkstaff(ROLES,R,O):-nl,write('check staff credentials for access.'),nl,
                    R=[SID,SECTION,PART|_],
                    write('  Resource='),write(R),nl,
                    write('  ROLES='),write(ROLES),nl,
                    write('  Operation='),write(O),nl,
                    ( ( (member(bursar,ROLES),SECTION==[section,gsi],O==read);
                       (member(bursar,ROLES),SECTION==[section,se],O==read);
                       (member(bursar,ROLES),SECTION==[section,se],O==save);
                       (member(bursar,ROLES),SECTION==[section,fa],O==read);
                       (member(bursar,ROLES),SECTION==[section,fa],O==save);
                       (member(bursar,ROLES),SECTION==[section,notes],O==read);
                       (member(academicadvisor,ROLES),SECTION==[section,gsi],O==read);
                       (member(academicadvisor,ROLES),SECTION==[section,ar],O==read);
                       (member(academicadvisor,ROLES),SECTION==[section,ar],O==save)
                      ) ->true(R,O)).


%**********UserTerminal1***********
user_terminal_readStudentRecord(AID,S)    :-
principal(AID,P),student(S),record_server_readStudentRecord(P,S).
user_terminal_saveStudentRecord(AID,S,SR)    :-
principal(AID,P),student(S),record_server_saveStudentRecord(P,S,SR).

%**********RecordServer2***********
%*** These facts represent student data in the record server ***
record_server_sr(acrespi).
record_server_sr(hblake).
record_server_gsi(acrespi,alex,crespi,indianapolis).
record_server_gsi(hblake,henry,blake,muncie).
record_server_note(acrespi,fburns, ['this is a note']).
record_server_note(acrespi,cwinchester,['this is another note']).

%*** Predicate for reading student record ***
record_server_readStudentRecord(P,S):-
   financial_server_readStudentRecord(P,S) ;
   academic_server_readStudentRecord(P,S) ;
   employment_server_readStudentRecord(P,S) ;
   (record_server_gsi(S,_,_,_),record_server_Guard(P,[[sr,S],[section,gsi],[all,all]],read)).

%*** Predicate for writing student record ***
record_server_saveStudentRecord(P,S,SR):-
   ((member(gsi,SR), record_server_Guard(P,[[sr,S],[section,gsi],[all,all]],save)) ; not member(gsi,SR) ),
   ((member(fa,SR) ,financial_server_saveStudentRecord(P,S,SR)); not member(fa,SR) ),
   ((member(ar,SR) ,academic_server_saveStudentRecord(P,S,SR)); not member(ar,SR) ),
   ((member(se,SR) ,employment_server_saveStudentRecord(P,S,SR)); not member(se,SR) ).
```

```
%*** Predicate for record server guard ***
record_server_Guard(P,R,O):- record_server_DynamicAttributes(P,NEWP,R),
  accessControl(NEWP,R,O).


%*** Predicate for record server dynamic attributes ***
record_server_DynamicAttributes(P,NEWP,R):-
  arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
  arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
  arg(3,P,GROUPPRED),
  (member([sr,AID],R)->append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
  (member([noteowner,AID],R)->append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
  NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).

%**********AcademicRecordServer2***********
%*** These facts represent student data in the academic record server ***
academic_server_ar(acrespi,dps([computer_science,business]),ts([[cs403,a],[cs595,b]])).
academic_server_ar(hblake,dps([chemistry]),ts([[c101,a],[c132,a]])).

%*** These predicates represent the academic record server ***
academic_server_readStudentRecord(P,S):-
academic_server_ar(S,_,_),academic_server_Guard(P,[[sr,S],[section,ar],[all,all]],read).
academic_server_saveStudentRecord(P,S,AR):-
academic_server_Guard(P,[[sr,S],[section,ar],[all,all]],save).

%*** Predicate for academic record server guard ***
academic_server_Guard(P,R,O):- academic_server_DynamicAttributes(P,NEWP,R),
  accessControl(NEWP,R,O).

%*** Predicate for academic record server dynamic attributes ***
academic_server_DynamicAttributes(P,NEWP,R):-
  arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
  arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
  arg(3,P,GROUPPRED),
  (member([sr,AID],R)->append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
  (member([noteowner,AID],R)->append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
  NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).

%**********FinancialAidServer2***********
%*** These facts represent student data in the financial aid server ***
financial_server_fa(acrespi,[bankofamerica,sallemay],[pell],[volleyball],[]).

financial_server_fa(hblake,[bankofamerica,sallemay],[pell],[volleyball],[]).

%*** These predicates represent the financial aid server ***
financial_server_readStudentRecord(P,S):-
financial_server_fa(S,_,_,_,_),financial_server_Guard(P,[[sr,S],[section,fa],[all,all]],read).
financial_server_saveStudentRecord(P,S,AR):- financial_server_Guard(P,[[sr,S],[section,fa],[all,all]],save).

%*** Predicate for academic financial aid guard ***
financial_server_Guard(P,R,O):- financial_server_DynamicAttributes(P,NEWP,R),
  accessControl(NEWP,R,O).

%*** Predicate for academic financial aid dynamic attributes ***
```

```
financial_server_DynamicAttributes(P,NEWP,R):-
   arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
   arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
   arg(3,P,GROUPPRED),
   (member([sr,AID],R)->append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
   (member([noteowner,AID],R)->append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
   NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).

%**********StudentEmploymentServer2***********
%*** These facts represent student data in the student employment server ***
employment_server_se(acrespi,[[fall2003],[spring2004],[fall2004]],[]).

employment_server_se(hblake,[[fall2003],[spring2004],[fall2004]],[]).

%*** These predicates represent the student employment server ***
employment_server_readStudentRecord(P,S):-
employment_server_se(S,_,_),employment_server_Guard(P,[[sr,S],[section,se],[all,all]],read).
employment_server_saveStudentRecord(P,S,AR):-
employment_server_Guard(P,[[sr,S],[section,se],[all,all]],save).

%*** Predicate for student employment server guard ***
employment_server_Guard(P,R,O):- employment_server_DynamicAttributes(P,NEWP,R),
   accessControl(NEWP,R,O).

%*** Predicate for student employment server dynamic attributes ***
employment_server_DynamicAttributes(P,NEWP,R):-
   arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
   arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
   arg(3,P,GROUPPRED),
   (member([sr,AID],R)->append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
   (member([noteowner,AID],R)->append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
   NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).
```

Appendix D: Case Study Component TLA Specifications

This appendix contains all of the TLA specifications for individual components, system user attributes, system commands, and the access control policy used in the case study.

System Constants

```
CONSTANTS Threads, ThreadStates, Students, Staff, Command
```

Principals, User Attributes and Commands

```
EXTENDS  TLC, FiniteSets
VARIABLES attributes, principals, command_set

User_Init == /\ attributes = { ( "accessid" :> {"acrespi"}  @@
                          "roles"    :> {"student"} @@
                          "groups"   :> {"student"}
                        ),
                        ( "accessid" :> {"hblake"}  @@
                          "roles"    :> {"student"} @@
                          "groups"   :> {"student"}
                        ),
                        ( "accessid" :> {"fburns"}  @@
                          "roles"    :> {"bursar"} @@
                          "groups"   :> {"staff"}
                        ),
                        ( "accessid" :> {"cwinchester"}  @@
                          "roles"    :> {"academicadvisor"} @@
                          "groups"   :> {"staff"}
                        )
                       }
           /\ principals = Students \union Staff

           /\ command_set = [p:attributes, c:{"load"}, r:Students, d:{[gsi|->"",
                                                                       ar|->"",
                                                                       fa|->"",
                                                                       sf|->""]}]
                    \cup
                    [p:attributes, c:{"save"}, r:Students, d:[gsi:{"y","n"},
                                                              ar:{"y","n"},
                                                              fa:{"y","n"},
                                                              sf:{"y","n"}]]

User_Inv == /\ command_set \in SUBSET [p:attributes, c:UTCommand, r:Students, d:[gsi:{"y","n"},
                                                  ar:{"y","n"},
                                                  fa:{"y","n"},
                                                  sf:{"y","n"}]
                                                  \cup
                                                  {[gsi|->"",
                                                    ar|->"",
                                                    fa|->"",
                                                    sf|->""]}]
        /\ principals = Students \union Staff
```

Access Control Policy Evaluator

```
PolicyEvaluator(a,r,o) == \/( /\ r[1]="gsi"
                              /\ o="save"
                              /\ "owner" \in a["roles"]
                            )
                          \/( /\ r[1]="gsi"
                              /\ o="read"
                              /\ ( \*\/ "student" \in a["groups"]
                                   \/ "staff" \in a["groups"]
                                 )
                            )
                          \/( /\ r[1]="ar"
                              /\ o="save"
                              /\ "academicadvisor" \in a["roles"]
                            )
                          \/( /\ r[1]="ar"
                              /\ o="read"
                              /\ ( \/ "owner" \in a["roles"]
                                   \/ "academicadvisor" \in a["roles"]
                                 )
                            )
                          \/( /\ r[1]="fa"
                              /\ o="save"
                              /\ "bursar" \in a["roles"]
                            )
                          \/( /\ r[1]="fa"
                              /\ o="read"
                              /\ ( \/ "owner" \in a["roles"]
                                   \/ "bursar" \in a["roles"]
                                 )
                            )
                          \/( /\ r[1]="se"
                              /\ o="save"
                              /\ "bursar" \in a["roles"]
                            )
                          \/( /\ r[1]="se"
                              /\ o="read"
                              /\ "bursar" \in a["roles"]
                            )
```

Channels

```
EXTENDS  TLC, FiniteSets
VARIABLES ut_rs_save, ut_rs_read, sr_ar_save, sr_ar_read, sr_fa_save, sr_fa_read, sr_se_save,
sr_se_read,

CH_Init == /\ ut_rs_save = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |-
>""]]
        /\ ut_rs_read = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ rs_ar_save = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ rs_ar_read = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ rs_fa_save = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ rs_fa_read = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ rs_se_save = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ rs_se_read = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
```

UserTerminal

```
EXTENDS  TLC, FiniteSets
VARIABLES ut_threads, ut_threads_data,

UT_Inv  == /\ ut_threads \in [Threads -> ThreadStates]

UT_Init == /\ ut_threads = [t \in Threads |-> "dormant"]
        /\ ut_threads_data =[t \in Threads |-> [p|->("":>""),c|->"",r|->"",d|->""]]


UT_Read_Thread(t) == /\ ut_threads[t]="dormant"
                  /\ Cardinality(command_set)#0
                  /\ ut_threads_data'=[ut_threads_data EXCEPT ![t]= CHOOSE x \in
                     command_set:x#[p|->("":>""),r|->"",c|->"",d|->""]]
                  /\ command_set' = command_set \ {ut_threads_data'[t]}
                  /\ ut_threads'=[ut_threads EXCEPT ![t]="ready"]
                  /\ UNCHANGED<<principals,attributes>>
                  /\ UNCHANGED<<ut_rs_save,ut_rs_read>>

UT_Send_Save(t) == /\ ut_threads[t]="ready"
                /\ ut_threads_data[t].c="save"
                /\ ut_threads'=[ut_threads EXCEPT ![t]="active"]
                /\ ut_rs_save[t].state = "ready"
                /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[state|->"sent",
                                              data|->ut_threads_data[t],
                                           message |->""]]
                /\ UNCHANGED<<ut_threads_data>>
                /\ UNCHANGED<<command_set,principals,attributes>>
                /\ UNCHANGED<<ut_rs_read>>


UT_Receive_Save(t) == /\ ut_threads[t]="active"
                   /\ ut_threads_data[t].c="save"
```

```
                    /\ ut_rs_save[t].state = "replied"
                    /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[state|->"ready",
                                                   data|->ut_threads_data[t],
                                                message |->""]]
                    /\ ut_threads'=[ut_threads EXCEPT ![t]="dormant"]
                    /\ UNCHANGED<<ut_threads_data>>
                    /\ UNCHANGED<<command_set,principals,attributes>>
                    /\ UNCHANGED<<ut_rs_read>>


UT_Send_Read(t) == /\ ut_threads[t]="ready"
                  /\ ut_threads_data[t].c="read"
                  /\ ut_rs_read[t].state = "ready"
                  /\ ut_threads'=[ut_threads EXCEPT ![t]="active"]
                  /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"sent",
                                                 data|->ut_threads_data[t],
                                              message |-> ( "gsi" :> "" @@
                                                            "ar" :> "" @@
                                                            "fa" :> "" @@
                                                            "se" :> "" ) ]]
                  /\ UNCHANGED<<ut_threads_data>>
                  /\ UNCHANGED<<command_set,principals,attributes>>
                  /\ UNCHANGED<<ut_rs_save>>


UT_Receive_Read(t) == /\ ut_threads[t]="active"
                    /\ ut_threads_data[t].c="read"
                    /\ ut_rs_read[t].state = "replied"
                    /\ ut_threads'=[ut_threads EXCEPT ![t]="dormant"]
                    /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"ready",
                                                   data|->ut_threads_data[t],
                                                message|->( "gsi" :> "" @@
                                                             "ar" :> "" @@
                                                             "fa" :> "" @@
                                                             "se" :> "" )  ]]
                    /\ UNCHANGED<<ut_threads_data>>
                    /\ UNCHANGED<<command_set,principals,attributes>>
                    /\ UNCHANGED<<ut_rs_save>>



Waiting == /\ \A t \in Threads: ut_threads[t]="dormant"
                  /\ Cardinality(command_set)=0
          /\ UNCHANGED<<ut_threads_data, ut_threads>>
          /\ UNCHANGED<<command_set,principals,attributes>>
          /\ UNCHANGED<<ut_rs_save,ut_rs_read>>


UserTerminal(t) == \/ UT_Send_Save(t)
                  \/ UT_Send_Read(t)
                  \/ UT_Receive_Save(t)
                  \/ UT_Receive_Read(t)
                  \/ UT_Read_Thread(t)
                  \/ Waiting
```

RecordServer

```
EXTENDS  TLC, FiniteSets
VARIABLES gsi

RS_Init == /\ gsi = ( "acrespi" :> << "Alex", "Crespi", "Indianapolis", "IN" >> @@
                 "hblake" :> << "Henry", "Blake", "Washington", "DC" >> @@
                 "ggeorge" :> << "Greg", "George", "Washington", "DC" >>    )

RS_Guard(a,r,o) == PolicyEvaluator(
                 [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
                 <<"gsi",r>>,
                 o         )

\* receive save command from UT and forward it along to each of the lower level servers
RS_Recv_Save(t) == /\ ut_rs_save[t].state = "sent"
              /\ rs_ar_save[t].state = "ready"
              /\ rs_fa_save[t].state = "ready"
              /\ rs_se_save[t].state = "ready"
              /\ rs_ar_save'=IF ut_rs_save[t].data.d.ar="y"
                        THEN [rs_ar_save EXCEPT ![t]=[state|->"sent",
                                          data|->ut_rs_save[t].data,
                                          message |->""]]
                        ELSE [rs_ar_save EXCEPT ![t]=[state|->"replied",
                                          data|->ut_rs_save[t].data,
                                          message |->"N/A"]]
              /\ rs_fa_save'=IF ut_rs_save[t].data.d.fa="y"
                        THEN [rs_fa_save EXCEPT ![t]=[state|->"sent",
                                          data|->ut_rs_save[t].data,
                                          message |->""]]
                        ELSE [rs_fa_save EXCEPT ![t]=[state|->"replied",
                                          data|->ut_rs_save[t].data,
                                          message |->"N/A"]]
              /\ rs_se_save'=IF ut_rs_save[t].data.d.se="y"
                        THEN [rs_se_save EXCEPT ![t]=[state|->"sent",
                                          data|->ut_rs_save[t].data,
                                          message |->""]]
                        ELSE [rs_se_save EXCEPT ![t]=[state|->"replied",
                                          data|->ut_rs_save[t].data,
                                          message |->"N/A"]]
              /\ UNCHANGED<<command_set,principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
              /\ UNCHANGED<<rs_ar_read>>
              /\ UNCHANGED<<rs_fa_read>>
              /\ UNCHANGED<<rs_se_read>>


RS_Save_Done(t) == /\ ut_rs_save[t].state = "sent"
              /\ rs_ar_save[t].state = "replied"
              /\ rs_fa_save[t].state = "replied"
              /\ rs_se_save[t].state = "replied"
              /\ rs_ar_save'=[rs_ar_save EXCEPT ![t]=[state|->"ready",
                                        data|->ut_rs_save[t].data,
                                        message |->""]]
              /\ rs_fa_save'=[rs_fa_save EXCEPT ![t]=[state|->"ready",data|->ut_rs_save[t].data,message |-
>""]]
              /\ rs_se_save'=[rs_se_save EXCEPT ![t]=[state|->"ready",data|->ut_rs_save[t].data,message |-
>""]]
              /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[   state|->"replied",
                                          data|->ut_threads_data[t],
                                          message |-> IF (/\ (\/ rs_ar_save[t].message="SAVED"
                                                            \/ rs_ar_save[t].message="N/A")
                                                         /\ (\/ rs_se_save[t].message="SAVED"
                                                            \/ rs_se_save[t].message="N/A")
                                                         /\ (\/ rs_fa_save[t].message="SAVED"
                                                            \/ rs_fa_save[t].message="N/A")
                                                         )
                                                    THEN "SAVED"
                                                    ELSE "BLOCKED"
                                        ]
                         ]
              /\ UNCHANGED<<command_set,principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ut_rs_read>>
              /\ UNCHANGED<<rs_ar_read>>
              /\ UNCHANGED<<rs_fa_read>>
              /\ UNCHANGED<<rs_se_read>>


RS_Recv_Read(t) == /\ ut_rs_read[t].state = "sent"
              /\ rs_ar_read[t].state = "ready"
              /\ rs_fa_read[t].state = "ready"
              /\ rs_se_read[t].state = "ready"
              /\ rs_ar_read'=[rs_ar_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
              /\ rs_fa_read'=[rs_fa_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
              /\ rs_se_read'=[rs_se_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
              /\ UNCHANGED<<command_set,principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
              /\ UNCHANGED<<rs_ar_save>>
              /\ UNCHANGED<<rs_fa_save>>
              /\ UNCHANGED<<rs_se_save>>


RS_Read_Done(t) == /\ ut_rs_read[t].state = "sent"
              /\ rs_ar_read[t].state = "replied"
              /\ rs_fa_read[t].state = "replied"
              /\ rs_se_read[t].state = "replied"
              /\ rs_ar_read'=[rs_ar_read EXCEPT ![t]=[state|->"ready",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
              /\ rs_fa_read'=[rs_fa_read EXCEPT ![t]=[state|->"ready",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
              /\ rs_se_read'=[rs_se_read EXCEPT ![t]=[state|->"ready",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
              /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[   state|->"replied",
                                          data|->ut_threads_data[t],
                                        message |->( "gsi" :> "READ" @@
                                                  "ar" :> rs_ar_read[t].message @@
                                                  "fa" :> rs_fa_read[t].message @@
                                                  "se" :> rs_se_read[t].message
                                                )
                                        ]
                         ]
              /\ UNCHANGED<<command_set,principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ut_rs_save>>
              /\ UNCHANGED<<rs_ar_save>>
              /\ UNCHANGED<<rs_fa_save>>
              /\ UNCHANGED<<rs_se_save>>


RecordServer(t) == \/ RS_Recv_Save(t)
              \/ RS_Recv_Read(t)
              \/ RS_Save_Done(t)
              \/ RS_Read_Done(t)
```

RecordServer2

```
EXTENDS  TLC, FiniteSets
VARIABLES gsi

RS_Init == /\ gsi = ( "acrespi" :> << "Alex", "Crespi", "Indianapolis", "IN" >> @@
                  "hblake" :> << "Henry", "Blake", "Washington", "DC" >> @@
                  "ggeorge" :> << "Greg", "George", "Washington", "DC" >>     )

RS_Guard(a,r,o) == PolicyEvaluator(
                  [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
                  <<"gsi",r>>,
                  o          )

\* receive save command from UT and forward it along to each of the lower level servers
RS_Recv_Save(t) == /\ ut_rs_save[t].state = "sent"
                /\ rs_ar_save[t].state = "ready"
                /\ rs_fa_save[t].state = "ready"
                /\ rs_se_save[t].state = "ready"
                /\ rs_ar_save'=IF ut_rs_save[t].data.d.ar="y"
```

```
                    THEN [rs_ar_save EXCEPT ![t]=[state|->"sent",
                                   data|->ut_rs_save[t].data,
                                   message |->""]]
                    ELSE [rs_ar_save EXCEPT ![t]=[state|->"replied",
                                   data|->ut_rs_save[t].data,
                                   message |->"N/A"]]
             /\ rs_fa_save'=IF ut_rs_save[t].data.d.fa="y"
                    THEN [rs_fa_save EXCEPT ![t]=[state|->"sent",
                                   data|->ut_rs_save[t].data,
                                   message |->""]]
                    ELSE [rs_fa_save EXCEPT ![t]=[state|->"replied",
                                   data|->ut_rs_save[t].data,
                                   message |->"N/A"]]
             /\ rs_se_save'=IF ut_rs_save[t].data.d.se="y"
                    THEN [rs_se_save EXCEPT ![t]=[state|->"sent",
                                   data|->ut_rs_save[t].data,
                                   message |->""]]
                    ELSE [rs_se_save EXCEPT ![t]=[state|->"replied",
                                   data|->ut_rs_save[t].data,
                                   message |->"N/A"]]
             /\ UNCHANGED<<command_set,principals,attributes>>
             /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
             /\ UNCHANGED<<gsi>>
             /\ UNCHANGED<<rs_ar_read>>
             /\ UNCHANGED<<rs_fa_read>>
             /\ UNCHANGED<<rs_se_read>>


RS_Save_Done(t) == /\ ut_rs_save[t].state = "sent"
             /\ rs_ar_save[t].state = "replied"
             /\ rs_fa_save[t].state = "replied"
             /\ rs_se_save[t].state = "replied"
             /\ rs_ar_save'=[rs_ar_save EXCEPT ![t]=[state|->"ready",
                                 data|->ut_rs_save[t].data,
                                 message |->""]]
             /\ rs_fa_save'=[rs_fa_save EXCEPT ![t]=[state|->"ready",data|->ut_rs_save[t].data,message |-
>""]]
             /\ rs_se_save'=[rs_se_save EXCEPT ![t]=[state|->"ready",data|->ut_rs_save[t].data,message |-
>""]]
             /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[   state|->"replied",
                                  data|->ut_threads_data[t],
                                  message |-> IF (/\ (\/ rs_ar_save[t].message="SAVED"
                                                  \/ rs_ar_save[t].message="N/A")
                                               /\ (\/ rs_se_save[t].message="SAVED"
                                                  \/ rs_se_save[t].message="N/A")
                                               /\ (\/ rs_fa_save[t].message="SAVED"
                                                  \/ rs_fa_save[t].message="N/A")
                                               /\ (\/ut_rs_save[t].data.d.gsi="n"
                                                  \/RS_Guard(ut_rs_save[t].data.p,
                                                        ut_rs_save[t].data.r,
                                                        ut_rs_save[t].data.c))
                                             )
                                        THEN "SAVED"
                                        ELSE "BLOCKED"
```

```
                                    ]
                         ]
                  /\ UNCHANGED<<command_set,principals,attributes>>
                  /\ UNCHANGED<<gsi>>
                  /\ UNCHANGED<<ut_rs_read>>
                  /\ UNCHANGED<<rs_ar_read>>
                  /\ UNCHANGED<<rs_fa_read>>
                  /\ UNCHANGED<<rs_se_read>>


RS_Recv_Read(t) == /\ ut_rs_read[t].state = "sent"
                  /\ rs_ar_read[t].state = "ready"
                  /\ rs_fa_read[t].state = "ready"
                  /\ rs_se_read[t].state = "ready"
                  /\ rs_ar_read'=[rs_ar_read EXCEPT ![t]=[state|->"sent",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                  /\ rs_fa_read'=[rs_fa_read EXCEPT ![t]=[state|->"sent",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                  /\ rs_se_read'=[rs_se_read EXCEPT ![t]=[state|->"sent",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                  /\ UNCHANGED<<command_set,principals,attributes>>
                  /\ UNCHANGED<<gsi>>
                  /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
                  /\ UNCHANGED<<rs_ar_save>>
                  /\ UNCHANGED<<rs_fa_save>>
                  /\ UNCHANGED<<rs_se_save>>


RS_Read_Done(t) == /\ ut_rs_read[t].state = "sent"
                  /\ rs_ar_read[t].state = "replied"
                  /\ rs_fa_read[t].state = "replied"
                  /\ rs_se_read[t].state = "replied"
                  /\ rs_ar_read'=[rs_ar_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                  /\ rs_fa_read'=[rs_fa_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                  /\ rs_se_read'=[rs_se_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                  /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[   state|->"replied",
                                      data|->ut_threads_data[t],
                                    message |->( "gsi" :> (IF RS_Guard(ut_rs_read[t].data.p,
                                                            ut_rs_read[t].data.r,
                                                            ut_rs_read[t].data.c)
                                                   THEN "READ"
                                                   ELSE "BLOCKED") @@
                                            "ar" :> rs_ar_read[t].message @@
                                            "fa" :> rs_fa_read[t].message @@
                                            "se" :> rs_se_read[t].message
```

```
                                        )
                                   ]
                         ]
                /\ UNCHANGED<<command_set,principals,attributes>>
                /\ UNCHANGED<<gsi>>
                /\ UNCHANGED<<ut_rs_save>>
                /\ UNCHANGED<<rs_ar_save>>
                /\ UNCHANGED<<rs_fa_save>>
                /\ UNCHANGED<<rs_se_save>>


RecordServer(t) == \/ RS_Recv_Save(t)
                \/ RS_Recv_Read(t)
                \/ RS_Save_Done(t)
                \/ RS_Read_Done(t)
```

RecordServer3

```
EXTENDS  TLC, FiniteSets
VARIABLES gsi, ar_cache, fa_cache, se_cache

RS_Init == /\ gsi = ( "acrespi" :> << "Alex", "Crespi", "Indianapolis", "IN" >> @@
                  "hblake" :> << "Henry", "Blake", "Washington", "DC" >> @@
                  "ggeorge" :> << "Greg", "George", "Washington", "DC" >>
                 )
          /\ ar_cache = [s \in Students |-> "MISS"]
          /\ fa_cache = [s \in Students |-> "MISS"]
          /\ se_cache = [s \in Students |-> "MISS"]

\* Guard operator for the record server.
RS_Guard(a,r,o) == PolicyEvaluator(
                  [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
                  <<"gsi",r>>,
                  o
                         )

\* receive save command from UT and forward it along to each of the lower level servers
RS_Recv_Save(t) == /\ ut_rs_save[t].state = "sent"
                /\ rs_ar_save[t].state = "ready"
                /\ rs_fa_save[t].state = "ready"
                /\ rs_se_save[t].state = "ready"
                /\ rs_ar_save'=IF ut_rs_save[t].data.d.ar="y"
                         THEN [rs_ar_save EXCEPT ![t]=[state|->"sent",
                                          data|->ut_rs_save[t].data,
                                          message |->""]]
                         ELSE [rs_ar_save EXCEPT ![t]=[state|->"replied",
                                          data|->ut_rs_save[t].data,
                                          message |->"N/A"]]
                /\ rs_fa_save'=IF ut_rs_save[t].data.d.fa="y"
                         THEN [rs_fa_save EXCEPT ![t]=[state|->"sent",
                                      data|->ut_rs_save[t].data,
                                      message |->""]]
                    ELSE [rs_fa_save EXCEPT ![t]=[state|->"replied",
                                      data|->ut_rs_save[t].data,
                                      message |->"N/A"]]
          /\ rs_se_save'=IF ut_rs_save[t].data.d.se="y"
                    THEN [rs_se_save EXCEPT ![t]=[state|->"sent",
                                      data|->ut_rs_save[t].data,
                                      message |->""]]
                    ELSE [rs_se_save EXCEPT ![t]=[state|->"replied",
                                      data|->ut_rs_save[t].data,
                                      message |->"N/A"]]
          /\ UNCHANGED<<command_set,principals,attributes>>
          /\ UNCHANGED<<gsi>>
          /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
          /\ UNCHANGED<<rs_ar_read>>
          /\ UNCHANGED<<rs_fa_read>>
          /\ UNCHANGED<<rs_se_read>>
          /\ UNCHANGED<<ar_cache,fa_cache,se_cache>>


RS_Save_Done(t) == /\ ut_rs_save[t].state = "sent"
          /\ rs_ar_save[t].state = "replied"
          /\ rs_fa_save[t].state = "replied"
          /\ rs_se_save[t].state = "replied"
          /\ rs_ar_save'=[rs_ar_save EXCEPT ![t]=[state|->"ready",
                                   data|->ut_rs_save[t].data,
                                   message |->""]]
          /\ rs_fa_save'=[rs_fa_save EXCEPT ![t]=[state|->"ready",
                                   data|->ut_rs_save[t].data,
                                   message |->""]]
          /\ rs_se_save'=[rs_se_save EXCEPT ![t]=[state|->"ready",
                                   data|->ut_rs_save[t].data,
                                   message |->""]]
          /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[   state|->"replied",
                                     data|->ut_threads_data[t],
                                   message |-> IF (/\ (\/ rs_ar_save[t].message="SAVED"
                                                      \/ rs_ar_save[t].message="N/A")
                                                   /\ (\/ rs_se_save[t].message="SAVED"
                                                      \/ rs_se_save[t].message="N/A")
                                                   /\ (\/ rs_fa_save[t].message="SAVED"
                                                      \/ rs_fa_save[t].message="N/A")
                                                   /\ (\/ut_rs_save[t].data.d.gsi="n"
                                                      \/RS_Guard(ut_rs_save[t].data.p,
                                                              ut_rs_save[t].data.r,
                                                              ut_rs_save[t].data.c))
                                                   )
                                               THEN "SAVED"
                                               ELSE "BLOCKED"
                                  ]
                    ]
          /\ ar_cache' =[ar_cache EXCEPT ![ut_rs_save[t].data.r]= IF rs_ar_save[t].message="SAVED"
                                                   THEN "HIT"
                                                   ELSE ar_cache[ut_rs_read[t].data.r] ]
```

```
                /\ fa_cache' =[fa_cache EXCEPT ![ut_rs_save[t].data.r]= IF rs_fa_save[t].message="SAVED"
                                                    THEN "HIT"
                                                    ELSE fa_cache[ut_rs_read[t].data.r] ]
                /\ se_cache' =[se_cache EXCEPT ![ut_rs_save[t].data.r]= IF rs_se_save[t].message="SAVED"
                                                    THEN "HIT"
                                                    ELSE se_cache[ut_rs_read[t].data.r] ]
                /\ UNCHANGED<<command_set,principals,attributes>>
                /\ UNCHANGED<<gsi>>
                /\ UNCHANGED<<ut_rs_read>>
                /\ UNCHANGED<<rs_ar_read>>
                /\ UNCHANGED<<rs_fa_read>>
                /\ UNCHANGED<<rs_se_read>>
                /\ UNCHANGED<<ar_cache,fa_cache,se_cache>>


RS_Recv_Read(t) == /\ ut_rs_read[t].state = "sent"
                /\ rs_ar_read[t].state = "ready"
                /\ rs_fa_read[t].state = "ready"
                /\ rs_se_read[t].state = "ready"

                /\ rs_ar_read'=IF ar_cache[ut_rs_read[t].data.r]="MISS"
                        THEN [rs_ar_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
                        ELSE [rs_ar_read EXCEPT ![t]=[state|->"replied",
                                        data|->ut_rs_read[t].data,
                                        message |->"READ"]]
                /\ rs_fa_read'=IF fa_cache[ut_rs_read[t].data.r]="MISS"
                        THEN [rs_fa_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
                        ELSE [rs_fa_read EXCEPT ![t]=[state|->"replied",
                                        data|->ut_rs_read[t].data,
                                        message |->"READ"]]
                /\ rs_se_read'=IF se_cache[ut_rs_read[t].data.r]="MISS"
                        THEN [rs_se_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
                        ELSE [rs_se_read EXCEPT ![t]=[state|->"replied",
                                        data|->ut_rs_read[t].data,
                                        message |->"READ"]]
                /\ UNCHANGED<<command_set,principals,attributes>>
                /\ UNCHANGED<<gsi>>
                /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
                /\ UNCHANGED<<rs_ar_save>>
                /\ UNCHANGED<<rs_fa_save>>
                /\ UNCHANGED<<rs_se_save>>
                /\ UNCHANGED<<ar_cache,fa_cache,se_cache>>


RS_Read_Done(t) == /\ ut_rs_read[t].state = "sent"
                /\ rs_ar_read[t].state = "replied"
                /\ rs_fa_read[t].state = "replied"
                /\ rs_se_read[t].state = "replied"
                /\ rs_ar_read'=[rs_ar_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                /\ rs_fa_read'=[rs_fa_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                /\ rs_se_read'=[rs_se_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
                /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"replied",
                                    data|->ut_threads_data[t],
                                    message |->( "gsi" :> (IF RS_Guard(ut_rs_read[t].data.p,
                                                              ut_rs_read[t].data.r,
                                                              ut_rs_read[t].data.c)
                                                        THEN "READ"
                                                        ELSE "BLOCKED") @@
                                                 "ar" :> rs_ar_read[t].message @@
                                                 "fa" :> rs_fa_read[t].message @@
                                                 "se" :> rs_se_read[t].message
                                               )
                                    ]
                           ]
                /\ ar_cache' =[ar_cache EXCEPT ![ut_rs_read[t].data.r]= IF rs_ar_read[t].message="READ"
                                                  THEN "HIT"
                                                  ELSE ar_cache[ut_rs_read[t].data.r] ]
                /\ fa_cache' =[fa_cache EXCEPT ![ut_rs_read[t].data.r]= IF rs_fa_read[t].message="READ"
                                                  THEN "HIT"
                                                  ELSE fa_cache[ut_rs_read[t].data.r] ]
                /\ se_cache' =[se_cache EXCEPT ![ut_rs_read[t].data.r]= IF rs_se_read[t].message="READ"
                                                  THEN "HIT"
                                                  ELSE se_cache[ut_rs_read[t].data.r] ]
                /\ UNCHANGED<<command_set,principals,attributes>>
                /\ UNCHANGED<<gsi>>
                /\ UNCHANGED<<ut_rs_save>>
                /\ UNCHANGED<<rs_ar_save>>
                /\ UNCHANGED<<rs_fa_save>>
                /\ UNCHANGED<<rs_se_save>>


RecordServer(t) == \/ RS_Recv_Save(t)
                \/ RS_Recv_Read(t)
                \/ RS_Save_Done(t)
                \/ RS_Read_Done(t)
```

RecordServer4

```
EXTENDS  TLC, FiniteSets
VARIABLES gsi, ar_cache, fa_cache, se_cache

RS_Init == /\ gsi = ( "acrespi" :> << "Alex", "Crespi", "Indianapolis", "IN" >> @@
                "hblake" :> << "Henry", "Blake", "Washington", "DC" >> @@
                "ggeorge" :> << "Greg", "George", "Washington", "DC" >>
               )
        /\ ar_cache = [s \in Students |-> "MISS"]
        /\ fa_cache = [s \in Students |-> "MISS"]
        /\ se_cache = [s \in Students |-> "MISS"]

\* Guard operator for the record server.
RS_Guard(a,r,o) == PolicyEvaluator(
                [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
                <<"gsi",r>>,
                o
                    )

\* receive save command from UT and forward it along to each of the lower level servers
RS_Recv_Save(t) == /\ ut_rs_save[t].state = "sent"
              /\ rs_ar_save[t].state = "ready"
              /\ rs_fa_save[t].state = "ready"
              /\ rs_se_save[t].state = "ready"
              /\ rs_ar_save'=IF ut_rs_save[t].data.d.ar="y"
                      THEN [rs_ar_save EXCEPT ![t]=[state|->"sent",
                                    data|->ut_rs_save[t].data,
                                    message |->""]]
                      ELSE [rs_ar_save EXCEPT ![t]=[state|->"replied",
                                    data|->ut_rs_save[t].data,
                                    message |->"N/A"]]
              /\ rs_fa_save'=IF ut_rs_save[t].data.d.fa="y"
                      THEN [rs_fa_save EXCEPT ![t]=[state|->"sent",
                                    data|->ut_rs_save[t].data,
                                    message |->""]]
                      ELSE [rs_fa_save EXCEPT ![t]=[state|->"replied",
                                    data|->ut_rs_save[t].data,
                                    message |->"N/A"]]
              /\ rs_se_save'=IF ut_rs_save[t].data.d.se="y"
                      THEN [rs_se_save EXCEPT ![t]=[state|->"sent",
                                    data|->ut_rs_save[t].data,
                                    message |->""]]
                      ELSE [rs_se_save EXCEPT ![t]=[state|->"replied",
                                    data|->ut_rs_save[t].data,
                                    message |->"N/A"]]
              /\ UNCHANGED<<command_set,principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
              /\ UNCHANGED<<rs_ar_read>>
              /\ UNCHANGED<<rs_fa_read>>
              /\ UNCHANGED<<rs_se_read>>
              /\ UNCHANGED<<ar_cache,fa_cache,se_cache>>



RS_Save_Done(t) == /\ ut_rs_save[t].state = "sent"
              /\ rs_ar_save[t].state = "replied"
              /\ rs_fa_save[t].state = "replied"
```

```
            /\ rs_se_save[t].state = "replied"
            /\ rs_ar_save'=[rs_ar_save EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_save[t].data,
                                    message |->""]]
            /\ rs_fa_save'=[rs_fa_save EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_save[t].data,
                                    message |->""]]
            /\ rs_se_save'=[rs_se_save EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_save[t].data,
                                    message |->""]]
            /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[state|->"replied",
                                    data|->ut_threads_data[t],
                                    message |-> IF (/\ (\/rs_ar_save[t].message="SAVED"
                                                        \/ rs_ar_save[t].message="N/A")
                                                     /\ (\/rs_se_save[t].message="SAVED"
                                                        \/ rs_se_save[t].message="N/A")
                                                     /\ (\/rs_fa_save[t].message="SAVED"
                                                        \/ rs_fa_save[t].message="N/A")
                                                     /\ (\/ut_rs_save[t].data.d.gsi="n"
                                                        \/RS_Guard(ut_rs_save[t].data.p,
                                                              ut_rs_save[t].data.r,
                                                              ut_rs_save[t].data.c))
                                                     )
                                                THEN "SAVED"
                                                ELSE "BLOCKED"
                                    ]
                      ]
            /\ ar_cache' =[ar_cache EXCEPT ![ut_rs_save[t].data.r]= IF rs_ar_save[t].message="SAVED"
                                                     THEN "HIT"
                                                     ELSE ar_cache[ut_rs_read[t].data.r] ]
            /\ fa_cache' =[fa_cache EXCEPT ![ut_rs_save[t].data.r]= IF rs_fa_save[t].message="SAVED"
                                                     THEN "HIT"
                                                     ELSE fa_cache[ut_rs_read[t].data.r] ]
            /\ se_cache' =[se_cache EXCEPT ![ut_rs_save[t].data.r]= IF rs_se_save[t].message="SAVED"
                                                     THEN "HIT"
                                                     ELSE se_cache[ut_rs_read[t].data.r] ]
            /\ UNCHANGED<<command_set,principals,attributes>>
            /\ UNCHANGED<<gsi>>
            /\ UNCHANGED<<ut_rs_read>>
            /\ UNCHANGED<<rs_ar_read>>
            /\ UNCHANGED<<rs_fa_read>>
            /\ UNCHANGED<<rs_se_read>>
            /\ UNCHANGED<<ar_cache,fa_cache,se_cache>>


RS_Recv_Read(t) == /\ ut_rs_read[t].state = "sent"
            /\ rs_ar_read[t].state = "ready"
            /\ rs_fa_read[t].state = "ready"
            /\ rs_se_read[t].state = "ready"

            /\ rs_ar_read'=IF /\ ar_cache[ut_rs_read[t].data.r]="HIT"
                              /\ RS_Guard(ut_rs_read[t].data.p,ut_rs_read[t].data.r,ut_rs_read[t].data.c)
                             THEN [rs_ar_read EXCEPT ![t]=[state|->"replied",
                                         data|->ut_rs_read[t].data,
                                        message |->"READ"]]
                    ELSE [rs_ar_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
            /\ rs_fa_read'=IF /\ fa_cache[ut_rs_read[t].data.r]="HIT"
                              /\ RS_Guard(ut_rs_read[t].data.p,ut_rs_read[t].data.r,ut_rs_read[t].data.c)
                    THEN [rs_fa_read EXCEPT ![t]=[state|->"replied",
                                        data|->ut_rs_read[t].data,
                                        message |->"READ"]]
                    ELSE [rs_fa_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
            /\ rs_se_read'=IF /\ se_cache[ut_rs_read[t].data.r]="HIT"
                              /\ RS_Guard(ut_rs_read[t].data.p,ut_rs_read[t].data.r,ut_rs_read[t].data.c)
                    THEN [rs_se_read EXCEPT ![t]=[state|->"replied",
                                        data|->ut_rs_read[t].data,
                                        message |->"READ"]]
                    ELSE [rs_se_read EXCEPT ![t]=[state|->"sent",
                                        data|->ut_rs_read[t].data,
                                        message |->""]]
            /\ UNCHANGED<<command_set,principals,attributes>>
            /\ UNCHANGED<<gsi>>
            /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
            /\ UNCHANGED<<rs_ar_save>>
            /\ UNCHANGED<<rs_fa_save>>
            /\ UNCHANGED<<rs_se_save>>
            /\ UNCHANGED<<ar_cache,fa_cache,se_cache>>


RS_Read_Done(t) == /\ ut_rs_read[t].state = "sent"
            /\ rs_ar_read[t].state = "replied"
            /\ rs_fa_read[t].state = "replied"
            /\ rs_se_read[t].state = "replied"
            /\ rs_ar_read'=[rs_ar_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
            /\ rs_fa_read'=[rs_fa_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
            /\ rs_se_read'=[rs_se_read EXCEPT ![t]=[state|->"ready",
                                    data|->ut_rs_read[t].data,
                                    message |->""]]
            /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"replied",
                                    data|->ut_threads_data[t],
                                    message |->( "gsi" :> (IF RS_Guard(ut_rs_read[t].data.p,
                                                                ut_rs_read[t].data.r,
                                                                ut_rs_read[t].data.c)
                                                      THEN "READ"
                                                      ELSE "BLOCKED") @@
                                                 "ar" :> rs_ar_read[t].message @@
                                                 "fa" :> rs_fa_read[t].message @@
                                                 "se" :> rs_se_read[t].message
                                               )
                                    ]
                    ]
            /\ ar_cache' =[ar_cache EXCEPT ![ut_rs_read[t].data.r]= IF rs_ar_read[t].message="READ"
                                                 THEN "HIT"
                                                 ELSE ar_cache[ut_rs_read[t].data.r] ]
            /\ fa_cache' =[fa_cache EXCEPT ![ut_rs_read[t].data.r]= IF rs_fa_read[t].message="READ"
                                                 THEN "HIT"
                                                 ELSE fa_cache[ut_rs_read[t].data.r] ]
            /\ se_cache' =[se_cache EXCEPT ![ut_rs_read[t].data.r]= IF rs_se_read[t].message="READ"
                                                 THEN "HIT"
                                                 ELSE se_cache[ut_rs_read[t].data.r] ]
            /\ UNCHANGED<<command_set,principals,attributes>>
            /\ UNCHANGED<<gsi>>
            /\ UNCHANGED<<ut_rs_save>>
            /\ UNCHANGED<<rs_ar_save>>
            /\ UNCHANGED<<rs_fa_save>>
            /\ UNCHANGED<<rs_se_save>>


RecordServer(t) == \/ RS_Recv_Save(t)
            \/ RS_Recv_Read(t)
            \/ RS_Save_Done(t)
            \/ RS_Read_Done(t)
```

AcademicRecordServer1

```
EXTENDS  TLC, FiniteSets
VARIABLES ar

AR_Init ==  ar = ( "acrespi" :> ( "majors" :>     <<"computer science","business">> @@
                        "transcript" :> << <<"cs403","a">>,<<"cs595","b">> >>
                      ) @@
            "hblake"  :> ( "majors" :>     <<"chemistry">> @@
                        "transcript" :> << <<"c101","a">>,<<"c132","a">>  >>
                      ) @@
            "ggeorge"  :> ( "majors" :>     <<"chemistry">> @@
                        "transcript" :> << <<"c101","a">>,<<"c132","a">>  >>
                      ))

\* Guard operator for the academic record server.
AR_Guard(a,r,o) == PolicyEvaluator(
            [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
            <<"ar",r>>,
            o          )

AR_Save(t) == /\ rs_ar_save[t].state = "sent"
        /\ rs_ar_save'=[rs_ar_save EXCEPT ![t]=[   state|->"replied",
                                       data|->rs_ar_save[t].data,
                                     message |->"SAVED"
                                     ]
                  ]
```

```
          /\ UNCHANGED<<command_set,principals,attributes>>
          /\ UNCHANGED<<ar>>
          /\ UNCHANGED<<rs_ar_read>>


AR_Read(t) == /\ rs_ar_read[t].state = "sent"
          /\ rs_ar_read'=[rs_ar_read EXCEPT ![t]=[   state|->"replied",
                                                data|->rs_ar_read[t].data,
                                              message |->"READ"
                                             ]
                    ]
          /\ UNCHANGED<<command_set,principals,attributes>>
          /\ UNCHANGED<<ar>>
          /\ UNCHANGED<<rs_ar_save>>

AR_Server(t) == \/ AR_Save(t)
           \/ AR_Read(t)
```

AcademicRecordServer2

```
EXTENDS  TLC, FiniteSets
VARIABLES ar

AR_Init ==  ar = ( "acrespi" :> ( "majors" :>    <<"computer science","business">> @@
                        "transcript" :> << <<"cs403","a">>,<<"cs595","b">> >>
                       ) @@
             "hblake"  :> ( "majors" :>    <<"chemistry">> @@
                        "transcript" :> << <<"c101","a">>,<<"c132","a">>  >>
                       ) @@
             "ggeorge"  :> ( "majors" :>    <<"chemistry">> @@
                        "transcript" :> << <<"c101","a">>,<<"c132","a">>  >>
                       )
             )

\* Guard operator for the academic record server.
AR_Guard(a,r,o) == PolicyEvaluator(
             [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
             <<"ar",r>>,
             o
                       )

AR_Save(t) == /\ sr_ar_save[t].state = "sent"
          /\ sr_ar_save'=[sr_ar_save EXCEPT ![t]=[   state|->"replied",
                                                data|->sr_ar_save[t].data,
                                              message |->(IF AR_Guard(sr_ar_save[t].data.p,
                                                             sr_ar_save[t].data.r,
                                                             sr_ar_save[t].data.c)
                                                     THEN "SAVED"
                                                     ELSE "BLOCKED")
                                             ]
                        ]
            /\ UNCHANGED<<command_set,principals,attributes>>
            /\ UNCHANGED<<ar>>
            /\ UNCHANGED<<sr_ar_read>>
            /\ UNCHANGED<<ar_cache,fa_cache,se_cache>>


AR_Read(t) == /\ sr_ar_read[t].state = "sent"
            /\ sr_ar_read'=[sr_ar_read EXCEPT ![t]=[   state|->"replied",
                                        data|->sr_ar_read[t].data,
                                     message |->(IF AR_Guard(sr_ar_read[t].data.p,
                                                    sr_ar_read[t].data.r,
                                                    sr_ar_read[t].data.c)
                                             THEN "READ"
                                             ELSE "BLOCKED")
                                     ]
                        ]
            /\ UNCHANGED<<command_set,principals,attributes>>
            /\ UNCHANGED<<ar>>
            /\ UNCHANGED<<sr_ar_save>>
            /\ UNCHANGED<<ar_cache,fa_cache,se_cache>>

AR_Server(t) == \/ AR_Save(t)
             \/ AR_Read(t)
```

FinancialAidServer1

```
EXTENDS  TLC, FiniteSets
VARIABLES fa

FA_Init ==    fa =( "acrespi" :> ( "loans" :>       <<"Bank of America","sallemay">> @@
                          "grants" :>      <<"pell">> @@
                          "scholarships" :> <<"volleyball">>
                         ) @@
                "hblake" :>  ( "loans" :>       <<>> @@
                          "grants" :>      <<>> @@
                          "scholarships" :> <<>>
                         ) @@
                "ggeorge" :>  ( "loans" :>       <<>> @@
                          "grants" :>      <<>> @@
                          "scholarships" :> <<>>
                         ) )

\* Guard operator for the student financial server.
FA_Guard(a,r,o) == PolicyEvaluator(
                [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
                <<"fa",r>>,
                o          )

FA_Save(t) == /\ rs_fa_save[t].state = "sent"
```

```
        /\ rs_fa_save'=[rs_fa_save EXCEPT ![t]=[   state|->"replied",
                                              data|->rs_fa_save[t].data,
                                            message |->"SAVED"
                                            ]
                   ]
        /\ UNCHANGED<<command_set,principals,attributes>>
        /\ UNCHANGED<<fa>>
        /\ UNCHANGED<<rs_fa_read>>


FA_Read(t) == /\ rs_fa_read[t].state = "sent"
        /\ rs_fa_read'=[rs_fa_read EXCEPT ![t]=[   state|->"replied",
                                              data|->rs_fa_read[t].data,
                                            message |->"READ"
                                            ]
                   ]
        /\ UNCHANGED<<command_set,principals,attributes>>
        /\ UNCHANGED<<fa>>
        /\ UNCHANGED<<rs_fa_save>>


FA_Server(t) == \/ FA_Save(t)
         \/ FA_Read(t)
```

FinancialAidServer2

```
EXTENDS  TLC, FiniteSets
VARIABLES fa

FA_Init ==    fa =( "acrespi" :> ( "loans" :>       <<"Bank of America","sallemay">> @@
                     "grants" :>      <<"pell">> @@
                     "scholarships" :> <<"volleyball">>
                    ) @@
           "hblake" :>  ( "loans" :>       <<>> @@
                     "grants" :>      <<>> @@
                     "scholarships" :> <<>>
                    ) @@
           "ggeorge" :>  ( "loans" :>       <<>> @@
                     "grants" :>      <<>> @@
                     "scholarships" :> <<>>
                    ) )

\* Guard operator for the student financial server.
FA_Guard(a,r,o) == PolicyEvaluator(
              [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
              <<"fa",r>>,
              o          )

FA_Save(t) == /\ rs_fa_save[t].state = "sent"
        /\ rs_fa_save'=[rs_fa_save EXCEPT ![t]=[   state|->"replied",
                                    data|->rs_fa_save[t].data,
                                  message |->(IF FA_Guard(rs_fa_save[t].data.p,
                                                  rs_fa_save[t].data.r,
                                                  rs_fa_save[t].data.c)
                                          THEN "SAVED"
                                          ELSE "BLOCKED")
                                  ]
                       ]
        /\ UNCHANGED<<command_set,principals,attributes>>
        /\ UNCHANGED<<fa>>
        /\ UNCHANGED<<rs_fa_read>>


FA_Read(t) == /\ rs_fa_read[t].state = "sent"
        /\ rs_fa_read'=[rs_fa_read EXCEPT ![t]=[   state|->"replied",
                                    data|->rs_fa_read[t].data,
                                  message |->(IF FA_Guard(rs_fa_read[t].data.p,
                                                  rs_fa_read[t].data.r,
                                                  rs_fa_read[t].data.c)
                                         THEN "READ"
                                         ELSE "BLOCKED")
                                  ]
                       ]
        /\ UNCHANGED<<command_set,principals,attributes>>
        /\ UNCHANGED<<fa>>
        /\ UNCHANGED<<rs_fa_save>>



FA_Server(t) == \/ FA_Save(t)
         \/ FA_Read(t)
```

StudentEmploymentServer1

```
EXTENDS  TLC, FiniteSets
VARIABLES se

SE_Init ==    se =( "acrespi" :> ("workstudy":><<"Fall2003","Spring2004","Fall2004">>) @@
              "hblake"  :> ("workstudy":><<"Fall2004">>) @@
              "ggeorge"  :> ("workstudy":><<"Fall2004">>)
             )

\* Guard operator for the financial management server.
SE_Guard(a,r,o) == PolicyEvaluator(
               [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
               <<"se",r>>,
               o          )

SE_Save(t) == /\ rs_se_save[t].state = "sent"
        /\ rs_se_save'=[rs_se_save EXCEPT ![t]=[    state|->"replied",
                                        data|->rs_se_save[t].data,
                                     message |->"SAVED
                                    ]
                ]
        /\ UNCHANGED<<command_set,principals,attributes>>
        /\ UNCHANGED<<se>>
        /\ UNCHANGED<<rs_se_read>>


SE_Read(t) == /\ rs_se_read[t].state = "sent"
        /\ rs_se_read'=[rs_se_read EXCEPT ![t]=[   state|->"replied",
                                        data|->rs_se_read[t].data,
                                     message |->"READ"
                                    ]
                ]
        /\ UNCHANGED<<command_set,principals,attributes>>
        /\ UNCHANGED<<se>>
        /\ UNCHANGED<<rs_se_save>>


SE_Server(t) == \/ SE_Save(t)
           \/ SE_Read(t)
```

StudentEmploymentServer2

```
EXTENDS  TLC, FiniteSets
VARIABLES se

SE_Init ==    se =( "acrespi" :> ("workstudy":><<"Fall2003","Spring2004","Fall2004">>) @@
              "hblake"  :> ("workstudy":><<"Fall2004">>) @@
              "ggeorge"  :> ("workstudy":><<"Fall2004">>)
             )

\* Guard operator for the financial management server.
SE_Guard(a,r,o) == PolicyEvaluator(
              [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
              <<"se",r>>,
              o          )

SE_Save(t) == /\ rs_se_save[t].state = "sent"
        /\ rs_se_save'=[rs_se_save EXCEPT ![t]=[    state|->"replied",
                                        data|->rs_se_save[t].data,
                                     message |->(IF SE_Guard(rs_se_save[t].data.p,
                                                  rs_se_save[t].data.r,
                                                  rs_se_save[t].data.c)
                                            THEN "SAVED"
                                            ELSE "BLOCKED")
                                    ]
                ]
        /\ UNCHANGED<<command_set,principals,attributes>>
```

```
            /\ UNCHANGED<<se>>
            /\ UNCHANGED<<rs_se_read>>


SE_Read(t) == /\ rs_se_read[t].state = "sent"
            /\ rs_se_read'=[rs_se_read EXCEPT ![t]=[   state|->"replied",
                                              data|->rs_se_read[t].data,
                                           message |->(IF SE_Guard(rs_se_read[t].data.p,
                                                          rs_se_read[t].data.r,
                                                          rs_se_read[t].data.c)
                                                   THEN "READ"
                                                   ELSE "BLOCKED")
                                          ]
                         ]
            /\ UNCHANGED<<command_set,principals,attributes>>
            /\ UNCHANGED<<se>>
            /\ UNCHANGED<<rs_se_save>>


SE_Server(t) == \/ SE_Save(t)
              \/ SE_Read(t)
```

System Specification

```
TypeInvariant == /\ User_Inv
               /\ UT_Inv


Init  == /\ CH_Init
        /\ UT_Init
        /\ User_Init
        /\ RS_Init
        /\ AR_Init
        /\ FA_Init
        /\ SE_Init


vars == <<ut_threads,ut_threads_data,
         command_set,principals,attributes,
         gsi,ar,fa,se,
         ut_rs_save,ut_rs_read,
         rs_ar_save,rs_ar_read,
         rs_fa_save,rs_fa_read,
         rs_se_save,rs_se_read,
         ar_cache,fa_cache,se_cache>>

Next  ==  \E t \in Threads : ( UserTerminal(t)\/RecordServer(t)\/AR_Server(t)\/FA_Server(t)\/SE_Server(t))
Liveness == WF_vars(Next)

Spec  ==  Init /\ [][Next]_vars /\ Liveness
```

Liveness Properties

```
\*******************************************************************
\*** The following statements represent the liveness properties   ***
\*** to be tested
\*******************************************************************

\*************************************************************************
\*Verify that students can read all parts of their own student record
\*************************************************************************
Start_Read_Self(t) == /\ ENABLED(UT_Send_Read(t))
                  /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]

End_Read_Self(t) == /\ ENABLED(UT_Receive_Read(t))
                /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                /\ ut_rs_read[t].message["gsi"]="READ"
                /\ ut_rs_read[t].message["ar"] ="READ"
                /\ ut_rs_read[t].message["fa"] ="READ"
                /\ ut_rs_read[t].message["se"] ="READ"

Self_Read ==\A t \in Threads: Start_Read_Self(t) ~> End_Read_Self(t)

\*************************************************************************
\* Verify that students can read NO part of other student's records
\*************************************************************************
Start_Read_Other_Student(t) == /\ ENABLED(UT_Send_Read(t))
                         /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                         /\ "student" \in ut_threads_data[t].p["groups"]

End_Read_Other_Student(t) == /\ ENABLED(UT_Receive_Read(t))
                        /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                        /\ "student" \in ut_threads_data[t].p["groups"]
                        /\ ut_rs_read[t].message["gsi"]="BLOCKED"
                        /\ ut_rs_read[t].message["ar"] ="BLOCKED"
                        /\ ut_rs_read[t].message["fa"] ="BLOCKED"
                        /\ ut_rs_read[t].message["se"] ="BLOCKED"

Other_Student_Read ==\A t \in Threads: Start_Read_Other_Student(t) ~> End_Read_Other_Student(t)


\*************************************************************************
\*student cannot save other student's records
\*************************************************************************
Start_Save_Other_Student(t) == /\ ENABLED(UT_Send_Save(t))
                         /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                         /\ ut_threads_data[t].d # [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]
                         /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Other_Student(t) == /\ ENABLED(UT_Receive_Save(t))
                          /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                          /\ ut_threads_data[t].d # [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]
                          /\ "student" \in ut_threads_data[t].p["groups"]
```

```
                              /\ ut_rs_save[t].message="BLOCKED"

Other_Student_Save ==\A t \in Threads: Start_Save_Other_Student(t) ~> End_Save_Other_Student(t)


\**************************************************************************
\*Students can only save their own gsi and no other portion of their student record.
\**************************************************************************
Start_Save_Self_GSI(t) == /\ ENABLED(UT_Send_Save(t))
                    /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                    /\ ut_threads_data[t].d = [gsi |->"y", ar |->"n", fa |->"n", se |->"n"]
                    /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Self_GSI(t) == /\ ENABLED(UT_Receive_Save(t))
                       /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                       /\ ut_threads_data[t].d = [gsi |->"y", a
r |->"n", fa |->"n", se |->"n"]
                       /\ "student" \in ut_threads_data[t].p["groups"]
                       /\ ut_rs_save[t].message="SAVED"

Start_Save_Self_notGSI(t) == /\ ENABLED(UT_Send_Save(t))
                    /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                    /\ ut_threads_data[t].d \in [gsi:{"y","n"}, ar:{"y","n"}, fa:{"y","n"}, se:{"y","n"}]
                                    \ {[gsi |->"y", ar |->"n", fa |->"n", se |->"n"],
                                      [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]}
                    /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Self_notGSI(t) == /\ ENABLED(UT_Receive_Save(t))
                    /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                    /\ ut_threads_data[t].d \in [gsi:{"y","n"}, ar:{"y","n"}, fa:{"y","n"}, se:{"y","n"}]
                                    \ {[gsi |->"y", ar |->"n", fa |->"n", se |->"n"],
                                      [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]}
                    /\ "student" \in ut_threads_data[t].p["groups"]
                       /\ ut_rs_save[t].message="BLOCKED"

Self_Save ==\A t \in Threads: /\ (Start_Save_Self_GSI(t) ~> End_Save_Self_GSI(t))
                          /\ (Start_Save_Self_notGSI(t) ~> End_Save_Self_notGSI(t))
```

Configuration File

```
(***************************************************************************)
(* TLC requires that every declared constant in the specification be       *)
(* assigned a value by a CONSTANT statement in the configuration file.     *)
(***************************************************************************)
CONSTANTS

 Threads    = {1,2}
 Students   = {"acrespi","hblake"}
 Staff      = {"fburns","cwinchster"}
 UTCommand    = {"save","read"}
 ThreadStates = {"dormant","ready","active"}


SPECIFICATION Spec
  \* This statement tells TLC that Spec is the specification to be
  \* checked.

INVARIANT TypeInvariant
  \* This statement tells TLC to check that TypeInvariant is an
  \* invariant of the specification.

PROPERTY Self_Read Other_Student_Read Other_Student_Save Self_Save
  \* This statement tells TLC which properties to check as the model is checked
  \* In this case, they are liveness properties.
```

Appendix E: Case Study Composed TLA Specifications

This appendix contains the TLA access control model for test system 2 in this thesis. The system protects all resources in all components.

```
------------------------- MODULE StudentRecordsNew ----------------------------

EXTENDS Naturals,TLC,Sequences,FiniteSets

CONSTANTS Threads,     \* Set of account numbers for the system
        ThreadStates,
        UTCommand,
        Students,
        Staff,
        AllowPrint


VARIABLES ut_threads, \* status of all threads in the user terminal component
        principals, \* set of all principals in the system

\* The following four variables represent the student record datastructures

        gsi,      \* general student information data for all students
        ar,       \* academic record information for all students
        fa,       \* financial aid information for all students
        se,       \* student employment information for all students


        command_set,     \* set of commands to present to the system
        ut_threads_data, \* function that holds data for the thread
        attributes,      \* function for static attributes for principals

        ut_rs_save,
        ut_rs_read,
        sr_ar_save,
        sr_ar_read,
        sr_fa_save,
        sr_fa_read,
        sr_se_save,
        sr_se_read
----------------------------------------------------------------------

\*******************************************************************
\*** Section: Access Control                          ***
\*******************************************************************
\*** This portion of the specification defines the access control ***
\*** policy and static security attributes. This allows for       ***
\*** testing the with various access control privilages.          ***
\*******************************************************************

\* This statement initializes the static security attributes for each principal.
\* This must be done manually for each student and staff member.
```

```
User_Init == /\ attributes = { ( "accessid" :> {"acrespi"}  @@
                          "roles"    :> {"student"} @@
                          "groups"   :> {"student"}
                        ),
                        ( "accessid" :> {"hblake"}  @@
                          "roles"    :> {"student"} @@
                          "groups"   :> {"student"}
                        ),
                        ( "accessid" :> {"fburns"}  @@
                          "roles"    :> {"bursar"} @@
                          "groups"   :> {"staff"}
                        ),
                        ( "accessid" :> {"cwinchester"}  @@
                          "roles"    :> {"academicadvisor"} @@
                          "groups"   :> {"staff"}
                        )
                       }
          /\ principals = Students \union Staff
          /\ command_set =  [p:attributes, c:{"read"}, r:Students, d:{[gsi|->"",ar|->"",fa|->"",se|->""]}]
                          \cup
                         [p:attributes, c:{"save"}, r:Students, d:[gsi:{"y","n"}, ar:{"y","n"}, fa:{"y","n"},
se:{"y","n"}]]


User_Inv == /\ command_set \in SUBSET [p:attributes, c:UTCommand, r:Students, d:[gsi:{"y","n"},
ar:{"y","n"}, fa:{"y","n"}, se:{"y","n"}]
                                                  \cup {[gsi|->"",ar|->"",fa|->"",se|->""]}]
           /\ principals = Students \union Staff

\* The following statements will represent the access control policy that will be
\* used to evaluate access privilages to various system commands. the resource "r"
\* is a tuple of the form <<[record portion],[studentid]>> which is used to identify
\* what part of the student record for which student is being accessed.

PolicyEvaluator(a,r,o) == \/( /\ r[1]="gsi"
                          /\ o="save"
                          /\ "owner" \in a["roles"]
                        )
                      \/( /\ r[1]="gsi"
                          /\ o="read"
                          /\ ( \/ "owner" \in a["roles"]
                               \/ "staff" \in a["groups"]
                             )
                        )
                      \/( /\ r[1]="ar"
                          /\ o="save"
                          /\ "academicadvisor" \in a["roles"]
                        )
                      \/( /\ r[1]="ar"
                          /\ o="read"
                          /\ ( \/ "owner" \in a["roles"]
                               \/ "academicadvisor" \in a["roles"]
                             )
```

```
                    )
                  \/( /\ r[1]="fa"
                     /\ o="save"
                     /\ "bursar" \in a["roles"]
                    )
                  \/( /\ r[1]="fa"
                     /\ o="read"
                     /\ ( \/ "owner" \in a["roles"]
                         \/ "bursar" \in a["roles"]
                        )
                    )
                  \/( /\ r[1]="se"
                     /\ o="save"
                     /\ "bursar" \in a["roles"]
                    )
                  \/( /\ r[1]="se"
                     /\ o="read"
                     /\ ( \/ "owner" \in a["roles"]
                         \/ "bursar" \in a["roles"]
                        )
                    )

\*******************************************************************
\***           end of access control representation           ***
\*******************************************************************

\*******************************************************************
\*** COMPONENT: user Terminal                                 ***
\*******************************************************************
\*** The following four statements represent the cashier terminal ***
\*** in this system. it connects the request data to the          ***
\*** transaction statements.                                  ***
\*******************************************************************

UT_Inv  == /\ ut_threads \in [Threads -> ThreadStates]

UT_Init == /\ ut_threads = [t \in Threads |-> "dormant"]
        /\ ut_threads_data =[t \in Threads |-> [p|->("":>""),c|->"",r|->"",d|->""]]


UT_Read_Thread(t) == /\ ut_threads[t]="dormant"
              /\ Cardinality(command_set)#0
              /\ ut_threads_data'=[ut_threads_data EXCEPT ![t]= CHOOSE x \in command_set:x#[p|-
>("":>""),r|->"",c|->"",d|->""]]
              /\ command_set' = command_set \ {ut_threads_data'[t]}
              /\ ut_threads'=[ut_threads EXCEPT ![t]="ready"]
              /\ UNCHANGED<<principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ar>>
              /\ UNCHANGED<<fa>>
              /\ UNCHANGED<<se>>
              /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
              /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
              /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
```

```
                /\ UNCHANGED<<sr_se_save,sr_se_read>>

UT_Send_Save(t) == /\ ut_threads[t]="ready"
              /\ ut_threads_data[t].c="save"
              /\ ut_threads'=[ut_threads EXCEPT ![t]="active"]
              /\ ut_rs_save[t].state = "ready"
              /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[state|->"sent",data|->ut_threads_data[t],message |-
>""]]
              /\ UNCHANGED<<ut_threads_data>>
              /\ UNCHANGED<<command_set,principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ar>>
              /\ UNCHANGED<<fa>>
              /\ UNCHANGED<<se>>
              /\ UNCHANGED<<ut_rs_read>>
              /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
              /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
              /\ UNCHANGED<<sr_se_save,sr_se_read>>

UT_Receive_Save(t) == /\ ut_threads[t]="active"
                /\ ut_threads_data[t].c="save"
                /\ ut_rs_save[t].state = "replied"
                /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[state|->"ready",data|->ut_threads_data[t],message |-
>""]]
                /\ ut_threads'=[ut_threads EXCEPT ![t]="dormant"]
                /\ UNCHANGED<<command_set,ut_threads_data>>
                /\ UNCHANGED<<principals,gsi,ar,fa,se,attributes >>
                /\
UNCHANGED<<ut_rs_read,sr_ar_save,sr_ar_read,sr_fa_save,sr_fa_read,sr_se_save,sr_se_read>>
                /\ UNCHANGED<<ut_threads_data>>
                /\ UNCHANGED<<command_set,principals,attributes>>
                /\ UNCHANGED<<gsi>>
                /\ UNCHANGED<<ar>>
                /\ UNCHANGED<<fa>>
                /\ UNCHANGED<<se>>
                /\ UNCHANGED<<ut_rs_read>>
                /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
                /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
                /\ UNCHANGED<<sr_se_save,sr_se_read>>


UT_Send_Read(t) == /\ ut_threads[t]="ready"
              /\ ut_threads_data[t].c="read"
              /\ ut_rs_read[t].state = "ready"
              /\ ut_threads'=[ut_threads EXCEPT ![t]="active"]
              /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"sent",data|->ut_threads_data[t],message |-> (
"gsi" :> "" @@
                                                              "ar" :> "" @@
                                                              "fa" :> "" @@
                                                              "se" :> "" ) ]]
              /\ UNCHANGED<<ut_threads_data>>
              /\ UNCHANGED<<command_set,principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ar>>
```

```
                /\ UNCHANGED<<fa>>
                /\ UNCHANGED<<se>>
                /\ UNCHANGED<<ut_rs_save>>
                /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
                /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
                /\ UNCHANGED<<sr_se_save,sr_se_read>>

UT_Receive_Read(t) == /\ ut_threads[t]="active"
                 /\ ut_threads_data[t].c="read"
                 /\ ut_rs_read[t].state = "replied"
                 /\ ut_threads'=[ut_threads EXCEPT ![t]="dormant"]
                 /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[state|->"ready",data|->ut_threads_data[t], message|-
>( "gsi" :> "" @@
                                                                        "ar" :> "" @@
                                                                        "fa" :> "" @@
                                                                        "se" :> "" )  ]]
                 /\ UNCHANGED<<ut_threads_data>>
                 /\ UNCHANGED<<command_set,principals,attributes>>
                 /\ UNCHANGED<<gsi>>
                 /\ UNCHANGED<<ar>>
                 /\ UNCHANGED<<fa>>
                 /\ UNCHANGED<<se>>
                 /\ UNCHANGED<<ut_rs_save>>
                 /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
                 /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
                 /\ UNCHANGED<<sr_se_save,sr_se_read>>



Waiting == /\ \A t \in Threads: ut_threads[t]="dormant"
             /\ Cardinality(command_set)=0
          /\ UNCHANGED<<ut_threads_data, ut_threads>>
          /\ UNCHANGED<<command_set,principals,attributes>>
          /\ UNCHANGED<<gsi>>
          /\ UNCHANGED<<ar>>
          /\ UNCHANGED<<fa>>
          /\ UNCHANGED<<se>>
          /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
          /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
          /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
          /\ UNCHANGED<<sr_se_save,sr_se_read>>


UserTerminal(t) == \/ UT_Send_Save(t)
                \/ UT_Send_Read(t)
                \/ UT_Receive_Save(t)
                \/ UT_Receive_Read(t)
                \/ UT_Read_Thread(t)
                \/ Waiting

\*******************************************************************
\***          end of user terminal representation          ***
\*******************************************************************
```

```
\*******************************************************************
\*** COMMUNICATIONS CHANNELS                              ***
\*******************************************************************
\*** Each communications channel will be represented as a shared  ***
\*** variable with state indicating who can read and set its      ***
\*** value.                                 ***
\*** CHANNEL VARIABLES:                               ***
\***         ut_rs_save                      ***
\***         ut_rs_read                      ***
\***         sr_ar_save                      ***
\***         sr_ar_read                      ***
\***         sr_fa_save                      ***
\***         sr_fa_read                      ***
\***         sr_se_save                      ***
\***         sr_se_read                      ***
\***                                      ***
\*** channel variables contain a record that contains both the    ***
\*** state, the thread data, and message                    ***
\*******************************************************************


\* each channel can handle multiple threads and is modeled as function
\* with domain being the set of threads and the range being a set of records.

CH_Init == /\ ut_rs_save = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |-
>""]]
        /\ ut_rs_read = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ sr_ar_save = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ sr_ar_read = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ sr_fa_save = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ sr_fa_read = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ sr_se_save = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]
        /\ sr_se_read = [t \in Threads |-> [state|->"ready", data |->[p|->("":>""),r|->"",c|->""], message |->""]]

\*******************************************************************
\***           end of communications channels              ***
\*******************************************************************

\*******************************************************************
\*** COMPONENT: record server                          ***
\*******************************************************************
\*** The following statements represent the processing of the     ***
\*** record server.
\*******************************************************************


RS_Init == /\ gsi = ( "acrespi" :> << "Alex", "Crespi", "Indianapolis", "IN" >> @@
                "hblake" :> << "Henry", "Blake", "Washington", "DC" >> @@
                "ggeorge" :> << "Greg", "George", "Washington", "DC" >>
               )

\* Guard operator for the record server.
RS_Guard(a,r,o) == PolicyEvaluator(
                [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
                <<"gsi",r>>,
                o
                        )

\* receive save command from UT and forward it along to each of the lower level servers
RS_Recv_Save(t) == /\ ut_rs_save[t].state = "sent"
              /\ sr_ar_save[t].state = "ready"
              /\ sr_fa_save[t].state = "ready"
              /\ sr_se_save[t].state = "ready"
              /\ sr_ar_save'=IF ut_rs_save[t].data.d.ar="y"
                        THEN [sr_ar_save EXCEPT ![t]=[state|->"sent",data|->ut_rs_save[t].data,message |-
>""]]
                        ELSE [sr_ar_save EXCEPT ![t]=[state|->"replied",data|->ut_rs_save[t].data,message
|->"N/A"]]
              /\ sr_fa_save'=IF ut_rs_save[t].data.d.fa="y"
                        THEN [sr_fa_save EXCEPT ![t]=[state|->"sent",data|->ut_rs_save[t].data,message |-
>""]]
                        ELSE [sr_fa_save EXCEPT ![t]=[state|->"replied",data|->ut_rs_save[t].data,message
|->"N/A"]]
              /\ sr_se_save'=IF ut_rs_save[t].data.d.se="y"
                        THEN [sr_se_save EXCEPT ![t]=[state|->"sent",data|->ut_rs_save[t].data,message |-
>""]]
                        ELSE [sr_se_save EXCEPT ![t]=[state|->"replied",data|->ut_rs_save[t].data,message
|->"N/A"]]
              /\ UNCHANGED<<ut_threads_data, ut_threads>>
              /\ UNCHANGED<<command_set,principals,attributes>>
              /\ UNCHANGED<<gsi>>
              /\ UNCHANGED<<ar>>
              /\ UNCHANGED<<fa>>
              /\ UNCHANGED<<se>>
              /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
              /\ UNCHANGED<<sr_ar_read>>
              /\ UNCHANGED<<sr_fa_read>>
              /\ UNCHANGED<<sr_se_read>>



RS_Save_Done(t) == /\ ut_rs_save[t].state = "sent"
              /\ sr_ar_save[t].state = "replied"
              /\ sr_fa_save[t].state = "replied"
              /\ sr_se_save[t].state = "replied"
              /\ sr_ar_save'=[sr_ar_save EXCEPT ![t]=[state|->"ready",data|->ut_rs_save[t].data,message |-
>""]]
              /\ sr_fa_save'=[sr_fa_save EXCEPT ![t]=[state|->"ready",data|->ut_rs_save[t].data,message |-
>""]]
              /\ sr_se_save'=[sr_se_save EXCEPT ![t]=[state|->"ready",data|->ut_rs_save[t].data,message |-
>""]]
              /\ ut_rs_save'=[ut_rs_save EXCEPT ![t]=[   state|->"replied",
                                        data|->ut_threads_data[t],
                                      message |-> IF (/\ (sr_ar_save[t].message="SAVED" \/
sr_ar_save[t].message="N/A")
```

```
                                              /\ (sr_se_save[t].message="SAVED" \/
sr_se_save[t].message="N/A")
                                              /\ (sr_fa_save[t].message="SAVED" \/
sr_fa_save[t].message="N/A")
                                              /\ (\/ut_rs_save[t].data.d.gsi="n"
                                                 \/RS_Guard(ut_rs_save[t].data.p,
                                                        ut_rs_save[t].data.r,
                                                        ut_rs_save[t].data.c))
                                              )
                                            THEN "SAVED"
                                            ELSE "BLOCKED"
                                      ]
                            ]
                /\ UNCHANGED<<ut_threads_data, ut_threads>>
                /\ UNCHANGED<<command_set,principals,attributes>>
                /\ UNCHANGED<<gsi>>
                /\ UNCHANGED<<ar>>
                /\ UNCHANGED<<fa>>
                /\ UNCHANGED<<se>>
                /\ UNCHANGED<<ut_rs_read>>
                /\ UNCHANGED<<sr_ar_read>>
                /\ UNCHANGED<<sr_fa_read>>
                /\ UNCHANGED<<sr_se_read>>


RS_Recv_Read(t) == /\ ut_rs_read[t].state = "sent"
                /\ sr_ar_read[t].state = "ready"
                /\ sr_fa_read[t].state = "ready"
                /\ sr_se_read[t].state = "ready"
                /\ sr_ar_read'=[sr_ar_read EXCEPT ![t]=[state|->"sent",data|->ut_rs_read[t].data,message |-
>""]]
                /\ sr_fa_read'=[sr_fa_read EXCEPT ![t]=[state|->"sent",data|->ut_rs_read[t].data,message |-
>""]]
                /\ sr_se_read'=[sr_se_read EXCEPT ![t]=[state|->"sent",data|->ut_rs_read[t].data,message |-
>""]]
                /\ UNCHANGED<<ut_threads_data, ut_threads>>
                /\ UNCHANGED<<command_set,principals,attributes>>
                /\ UNCHANGED<<gsi>>
                /\ UNCHANGED<<ar>>
                /\ UNCHANGED<<fa>>
                /\ UNCHANGED<<se>>
                /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
                /\ UNCHANGED<<sr_ar_save>>
                /\ UNCHANGED<<sr_fa_save>>
                /\ UNCHANGED<<sr_se_save>>


RS_Read_Done(t) == /\ ut_rs_read[t].state = "sent"
                /\ sr_ar_read[t].state = "replied"
                /\ sr_fa_read[t].state = "replied"
                /\ sr_se_read[t].state = "replied"
                /\ sr_ar_read'=[sr_ar_read EXCEPT ![t]=[state|->"ready",data|->ut_rs_read[t].data,message |-
>""]]
```

```
             /\ sr_fa_read'=[sr_fa_read EXCEPT ![t]=[state|->"ready",data|->ut_rs_read[t].data,message |-
>""]]
             /\ sr_se_read'=[sr_se_read EXCEPT ![t]=[state|->"ready",data|->ut_rs_read[t].data,message |-
>""]]
             /\ ut_rs_read'=[ut_rs_read EXCEPT ![t]=[   state|->"replied",
                                        data|->ut_threads_data[t],
                                      message |->( "gsi" :> (IF RS_Guard(ut_rs_read[t].data.p,
                                                                ut_rs_read[t].data.r,
                                                                ut_rs_read[t].data.c)
                                                     THEN "READ"
                                                     ELSE "BLOCKED") @@
                                                "ar" :> sr_ar_read[t].message @@
                                                "fa" :> sr_fa_read[t].message @@
                                                "se" :> sr_se_read[t].message
                                               )
                                      ]
                          ]
             /\ UNCHANGED<<ut_threads_data, ut_threads>>
             /\ UNCHANGED<<command_set,principals,attributes>>
             /\ UNCHANGED<<gsi>>
             /\ UNCHANGED<<ar>>
             /\ UNCHANGED<<fa>>
             /\ UNCHANGED<<se>>
             /\ UNCHANGED<<ut_rs_save>>
             /\ UNCHANGED<<sr_ar_save>>
             /\ UNCHANGED<<sr_fa_save>>
             /\ UNCHANGED<<sr_se_save>>


RecordServer(t) == \/ RS_Recv_Save(t)
             \/ RS_Recv_Read(t)
             \/ RS_Save_Done(t)
             \/ RS_Read_Done(t)


\*****************************************************************
\***          end of rs                          ***
\*****************************************************************

\*****************************************************************
\*** COMPONENTS: lower lever servers                   ***
\*****************************************************************
\*** The following statements represent the processing of the     ***
\*** lowest level servers.
\*****************************************************************

AR_Init ==  ar = ( "acrespi" :> ( "majors" :>     <<"computer science","business">> @@
                              "transcript" :> << <<"cs403","a">>,<<"cs595","b">> >>
                            ) @@
             "hblake"  :> ( "majors" :>     <<"chemistry">> @@
                              "transcript" :> << <<"c101","a">>,<<"c132","a">>  >>
                            ) @@
             "ggeorge"  :> ( "majors" :>     <<"chemistry">> @@
                              "transcript" :> << <<"c101","a">>,<<"c132","a">>  >>
```

```
                        )
                )

\* Guard operator for the academic record server.
AR_Guard(a,r,o) == PolicyEvaluator(
            [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
            <<"ar",r>>,
            o
                    )

AR_Save(t) == /\ sr_ar_save[t].state = "sent"
        /\ sr_ar_save'=[sr_ar_save EXCEPT ![t]=[   state|->"replied",
                                data|->sr_ar_save[t].data,
                              message |->(IF AR_Guard(sr_ar_save[t].data.p,
                                                sr_ar_save[t].data.r,
                                                sr_ar_save[t].data.c)
                                      THEN "SAVED"
                                      ELSE "BLOCKED")
                              ]
                ]
        /\ UNCHANGED<<ut_threads_data, ut_threads>>
        /\ UNCHANGED<<command_set,principals,attributes>>
        /\ UNCHANGED<<gsi>>
        /\ UNCHANGED<<ar>>
        /\ UNCHANGED<<fa>>
        /\ UNCHANGED<<se>>
        /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
        /\ UNCHANGED<<sr_ar_read>>
        /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
        /\ UNCHANGED<<sr_se_save,sr_se_read>>


AR_Read(t) == /\ sr_ar_read[t].state = "sent"
        /\ sr_ar_read'=[sr_ar_read EXCEPT ![t]=[   state|->"replied",
                                data|->sr_ar_read[t].data,
                              message |->(IF AR_Guard(sr_ar_read[t].data.p,
                                                sr_ar_read[t].data.r,
                                                sr_ar_read[t].data.c)
                                      THEN "READ"
                                      ELSE "BLOCKED")
                              ]
                ]
        /\ UNCHANGED<<ut_threads_data, ut_threads>>
        /\ UNCHANGED<<command_set,principals,attributes>>
        /\ UNCHANGED<<gsi>>
        /\ UNCHANGED<<ar>>
        /\ UNCHANGED<<fa>>
        /\ UNCHANGED<<se>>
        /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
        /\ UNCHANGED<<sr_ar_save>>
        /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
        /\ UNCHANGED<<sr_se_save,sr_se_read>>
```

```
AR_Server(t) == \/ AR_Save(t)
           \/ AR_Read(t)



FA_Init ==    fa =( "acrespi" :> ( "loans" :>       <<"Bank of America","sallemay">> @@
                          "grants" :>       <<"pell">> @@
                          "scholarships" :> <<"volleyball">>
                         ) @@
                "hblake" :>  ( "loans" :>       <<>> @@
                          "grants" :>       <<>> @@
                          "scholarships" :> <<>>
                         ) @@
                "ggeorge" :>  ( "loans" :>       <<>> @@
                          "grants" :>       <<>> @@
                          "scholarships" :> <<>>
                         )
              )

\* Guard operator for the student financial server.
FA_Guard(a,r,o) == PolicyEvaluator(
                [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
                <<"fa",r>>,
                o
                         )

FA_Save(t) == /\ sr_fa_save[t].state = "sent"
          /\ sr_fa_save'=[sr_fa_save EXCEPT ![t]=[   state|->"replied",
                                       data|->sr_fa_save[t].data,
                                     message |->(IF FA_Guard(sr_fa_save[t].data.p,
                                                    sr_fa_save[t].data.r,
                                                    sr_fa_save[t].data.c)
                                            THEN "SAVED"
                                            ELSE "BLOCKED")
                                     ]
                       ]
          /\ UNCHANGED<<ut_threads_data, ut_threads>>
          /\ UNCHANGED<<command_set,principals,attributes>>
          /\ UNCHANGED<<gsi>>
          /\ UNCHANGED<<ar>>
          /\ UNCHANGED<<fa>>
          /\ UNCHANGED<<se>>
          /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
          /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
          /\ UNCHANGED<<sr_fa_read>>
          /\ UNCHANGED<<sr_se_save,sr_se_read>>


FA_Read(t) == /\ sr_fa_read[t].state = "sent"
           /\ sr_fa_read'=[sr_fa_read EXCEPT ![t]=[   state|->"replied",
                                       data|->sr_fa_read[t].data,
                                     message |->(IF FA_Guard(sr_fa_read[t].data.p,
                                                    sr_fa_read[t].data.r,
                                                    sr_fa_read[t].data.c)
                                              THEN "READ"
                                              ELSE "BLOCKED")
                                     ]
                       ]
          /\ UNCHANGED<<ut_threads_data, ut_threads>>
          /\ UNCHANGED<<command_set,principals,attributes>>
          /\ UNCHANGED<<gsi>>
          /\ UNCHANGED<<ar>>
          /\ UNCHANGED<<fa>>
          /\ UNCHANGED<<se>>
          /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
          /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
          /\ UNCHANGED<<sr_fa_save>>
          /\ UNCHANGED<<sr_se_save,sr_se_read>>


FA_Server(t) == \/ FA_Save(t)
              \/ FA_Read(t)



SE_Init ==    se =( "acrespi" :> ("workstudy":><<"Fall2003","Spring2004","Fall2004">>) @@
                "hblake"  :> ("workstudy":><<"Fall2004">>) @@
                "ggeorge"  :> ("workstudy":><<"Fall2004">>)
              )

\* Guard operator for the financial management server.
SE_Guard(a,r,o) == PolicyEvaluator(
                [ a EXCEPT !["roles"]=IF (r \in a["accessid"]) THEN a["roles"] \cup {"owner"} ELSE
a["roles"]],
                <<"se",r>>,
                o
                          )

SE_Save(t) == /\ sr_se_save[t].state = "sent"
          /\ sr_se_save'=[sr_se_save EXCEPT ![t]=[    state|->"replied",
                                       data|->sr_se_save[t].data,
                                     message |->(IF SE_Guard(sr_se_save[t].data.p,
                                                    sr_se_save[t].data.r,
                                                    sr_se_save[t].data.c)
                                            THEN "SAVED"
                                            ELSE "BLOCKED")
                                     ]
                      ]
          /\ UNCHANGED<<ut_threads_data, ut_threads>>
          /\ UNCHANGED<<command_set,principals,attributes>>
          /\ UNCHANGED<<gsi>>
          /\ UNCHANGED<<ar>>
          /\ UNCHANGED<<fa>>
          /\ UNCHANGED<<se>>
          /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
          /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
          /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
```

```
            /\ UNCHANGED<<sr_se_read>>


SE_Read(t) == /\ sr_se_read[t].state = "sent"
            /\ sr_se_read'=[sr_se_read EXCEPT ![t]=[   state|->"replied",
                                        data|->sr_se_read[t].data,
                                      message |->(IF SE_Guard(sr_se_read[t].data.p,
                                                    sr_se_read[t].data.r,
                                                    sr_se_read[t].data.c)
                                              THEN "READ"
                                              ELSE "BLOCKED")
                                      ]
                        ]
            /\ UNCHANGED<<ut_threads_data, ut_threads>>
            /\ UNCHANGED<<command_set,principals,attributes>>
            /\ UNCHANGED<<gsi>>
            /\ UNCHANGED<<ar>>
            /\ UNCHANGED<<fa>>
            /\ UNCHANGED<<se>>
            /\ UNCHANGED<<ut_rs_save,ut_rs_read>>
            /\ UNCHANGED<<sr_ar_save,sr_ar_read>>
            /\ UNCHANGED<<sr_fa_save,sr_fa_read>>
            /\ UNCHANGED<<sr_se_save>>


SE_Server(t) == \/ SE_Save(t)
             \/ SE_Read(t)


\*********************************************************************
\*** The following statements represent the liveness properties   ***
\*** to be tested
\*********************************************************************

\*Verify that students can read all parts of their own student record
Start_Read_Self(t) == /\ ENABLED(UT_Send_Read(t))
                  /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]

End_Read_Self(t) == /\ ENABLED(UT_Receive_Read(t))
                 /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                 /\ ut_rs_read[t].message["gsi"]="READ"
                 /\ ut_rs_read[t].message["ar"] ="READ"
                 /\ ut_rs_read[t].message["fa"] ="READ"
                 /\ ut_rs_read[t].message["se"] ="READ"

Self_Read ==\A t \in Threads: Start_Read_Self(t) ~> End_Read_Self(t)

\***************************************************************************
\* Verify that students can read NO part of other student's records
\***************************************************************************
Start_Read_Other_Student(t) == /\ ENABLED(UT_Send_Read(t))
                         /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                         /\ "student" \in ut_threads_data[t].p["groups"]
```

```
End_Read_Other_Student(t) == /\ ENABLED(UT_Receive_Read(t))
                        /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                        /\ "student" \in ut_threads_data[t].p["groups"]
                        /\ ut_rs_read[t].message["gsi"]="BLOCKED"
                        /\ ut_rs_read[t].message["ar"] ="BLOCKED"
                        /\ ut_rs_read[t].message["fa"] ="BLOCKED"
                        /\ ut_rs_read[t].message["se"] ="BLOCKED"

Other_Student_Read ==\A t \in Threads: Start_Read_Other_Student(t) ~> End_Read_Other_Student(t)


\*************************************************************************
\*student cannot save other student's records
\*************************************************************************
Start_Save_Other_Student(t) == /\ ENABLED(UT_Send_Save(t))
                         /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                         /\ ut_threads_data[t].d # [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]
                         /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Other_Student(t) == /\ ENABLED(UT_Receive_Save(t))
                          /\ \lnot(ut_threads_data[t].r \in ut_threads_data[t].p["accessid"])
                          /\ ut_threads_data[t].d # [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]
                          /\ "student" \in ut_threads_data[t].p["groups"]
                          /\ ut_rs_save[t].message="BLOCKED"

Other_Student_Save ==\A t \in Threads: Start_Save_Other_Student(t) ~> End_Save_Other_Student(t)


\*************************************************************************
\*Students can only save their own gsi and no other portion of their student record.
\*************************************************************************
Start_Save_Self_GSI(t) == /\ ENABLED(UT_Send_Save(t))
                    /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                    /\ ut_threads_data[t].d = [gsi |->"y", ar |->"n", fa |->"n", se |->"n"]
                    /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Self_GSI(t) == /\ ENABLED(UT_Receive_Save(t))
                      /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                      /\ ut_threads_data[t].d = [gsi |->"y", ar |->"n", fa |->"n", se |->"n"]
                      /\ "student" \in ut_threads_data[t].p["groups"]
                      /\ ut_rs_save[t].message="SAVED"

Start_Save_Self_notGSI(t) == /\ ENABLED(UT_Send_Save(t))
                    /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
                    /\ ut_threads_data[t].d \in [gsi:{"y","n"}, ar:{"y","n"}, fa:{"y","n"}, se:{"y","n"}]
                                    \ {[gsi |->"y", ar |->"n", fa |->"n", se |->"n"],
                                       [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]}
                    /\ "student" \in ut_threads_data[t].p["groups"]


End_Save_Self_notGSI(t) == /\ ENABLED(UT_Receive_Save(t))
                    /\ ut_threads_data[t].r \in ut_threads_data[t].p["accessid"]
```

```
                /\ ut_threads_data[t].d \in [gsi:{"y","n"}, ar:{"y","n"}, fa:{"y","n"}, se:{"y","n"}]
                                    \ {[gsi |->"y", ar |->"n", fa |->"n", se |->"n"],
                                       [gsi |->"n", ar |->"n", fa |->"n", se |->"n"]}
                /\ "student" \in ut_threads_data[t].p["groups"]
                  /\ ut_rs_save[t].message="BLOCKED"

Self_Save ==\A t \in Threads: /\ (Start_Save_Self_GSI(t) ~> End_Save_Self_GSI(t))
                      /\ (Start_Save_Self_notGSI(t) ~> End_Save_Self_notGSI(t))


\*****************************************************************
\*** The following statements represent the initialization of     ***
\*** the system and also the requests that are made from the      ***
\*** cashier terminal.                                       ***
\*****************************************************************

\*****************************************************************
\*** The following statements represent the type invariant of     ***
\*** the system.                                            ***
\*****************************************************************

TypeInvariant == /\ User_Inv
             /\ UT_Inv


Init  == /\ CH_Init
      /\ UT_Init
      /\ User_Init
      /\ RS_Init
      /\ AR_Init
      /\ FA_Init
      /\ SE_Init


vars == <<ut_threads,ut_threads_data,
        command_set,principals,attributes,
        gsi,ar,fa,se,
        ut_rs_save,ut_rs_read,
        sr_ar_save,sr_ar_read,
        sr_fa_save,sr_fa_read,
        sr_se_save,sr_se_read>>

Next  ==  \E t \in Threads : ( UserTerminal(t)\/RecordServer(t)\/AR_Server(t)\/FA_Server(t)\/SE_Server(t))

Liveness == WF_vars(Next)

Spec  ==  Init /\ [][Next]_vars /\ Liveness

-------------------------------------------------------------------

=======================================================================
```

Appendix F: Case Study Component Implementations

This appendix contains the Java implementation of the UserTerminal, RecordServer2, AcademicRecordServer2, FinancialAidServer2, StudentEmploymentServer2, and the Security Center distributed components. In addition, the program for running system tests is included along with all of the remote interfaces used in the system.

UserTerminal.java

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.net.MalformedURLException;
import java.text.NumberFormat;
import java.io.*;
import java.util.*;

/**
 * This class provides the implementation for the component
 * UserTerminal in the univerity domain example.
 *
 * @author Alex Crespi
 * @date October 2004
 * @version 1.0
 */
public class UserTerminal extends UnicastRemoteObject implements IStudentRecord, IConfigure
{
   private IStudentRecord studentRecordServer;
   private ISecurityCenter securityCenter;

   private String studentRecordServer_ID;
   private String securityCenter_ID;

   private String ummSpecURL;
   private PrintWriter outputFile;

   /**
    * Constructor.
    */
   public UserTerminal() throws RemoteException
   {
      reset_configuration();

         //open output trace files
         try{
                 outputFile=new PrintWriter(new FileWriter(new File("UserTerminalTrace.txt")));
```

```
            }catch(Exception e){
                    System.out.println("ERROR:"+e);
            }
    }

    /**
     * This method writes log information about the functioning of this component to a file
     *
     * @param student accessid for the student record being accessed
     * @param sal security attribute for the principal excuting the request
     * @param command the request being made
     * @param comment and information to record. typicall success or failure
     */
    public void writeTrace(String student,SecurityAttributeList sal, String command, String comment){

            String user = "";

            //get user accessid from security attribute list
            for(int k=0; k<sal.size(); k++){
                    SecurityAttribute sa=sal.getAttribute(k);
                    if (sa.attributeType.equals("accessid")){
                            ArrayList value=(ArrayList)sa.value;
                            user=(String)((ArrayList)value).get(0);
                    }
            }

            Date d=new Date();
            String time = (new Long(d.getTime())).toString();

            try{
                    outputFile.println(time+" ; UserTerminal ; "+user+" ; "+command+" ; "+ student +" ;
"+comment);
                    outputFile.flush();
            }catch(Exception e){
                    System.out.println("ERROR:"+e);
            }
    }


    /**
     * This method configures a component to work in a bank system.
     *
     * @param server_name IP[:port]/binding_name
     * @param option The values for this parameter is one of TransactionServerManager,
     * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
     */
    public void configure(String server_name, String option) throws RemoteException
    {
      String url;

      if(server_name == null || server_name.trim().equals(""))
        return;

      try
    {
      if(option.equals("StudentRecordServer"))
      {
        url = "rmi://" + server_name.trim();
        studentRecordServer= (IStudentRecord)Naming.lookup(url);
        studentRecordServer_ID = server_name.trim();
        System.out.println(option + ":" + server_name);
      }
      else if(option.equals("SecurityCenter"))
      {
        url = "rmi://" + server_name.trim();
        securityCenter= (ISecurityCenter)Naming.lookup(url);
        securityCenter_ID = server_name.trim();
        System.out.println(option + ":" + server_name);
      }
    }
    catch(MalformedURLException e)
    {
      System.out.println (e);
    }
    catch(NotBoundException e)
    {
      System.out.println (e);
    }
    catch(ConnectException e)
    {
      System.out.println(e);
    }
    catch(RemoteException e)
    {
      System.out.println (e);
    }
  }

  /**
   * This method deconfigures a component.
   *
   * @param server_name IP[:port]/binding_name
   * @param option The values for this parameter is one of TransactionServerManager,
   * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
   */
  public void deconfigure(String server_name, String option) throws RemoteException
  {

    if(server_name == null || server_name.trim().equals(""))
      return;

    if(option.equals("StudentRecordServer"))
    {
        studentRecordServer = null;
        studentRecordServer_ID = null;
    }
    else if(option.equals("SecurityCenter"))
    {
        securityCenter = null;
        securityCenter_ID = null;
    }
  }

  /**
   * This method returns the configuration of a component.
   *
   * @param option The values for this parameter is one of TransactionServerManager,
   * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
   */
  public Object list_configuration(String option) throws RemoteException
  {
    if(option == null || option.trim().equals(""))
    {
      return null;
    }
    else if(option.equals("StudentRecordServer"))
    {
      return studentRecordServer_ID;
    }
    else if(option.equals("SecurityCenter"))
    {
      return securityCenter_ID;
    }
    else
      return null;
  }

  /**
   * This method resets the configuration for a component.
   */
  public void reset_configuration() throws RemoteException
  {
    studentRecordServer= null;
    securityCenter = null;
    studentRecordServer_ID = null;
    securityCenter_ID = null;
  }

  /**
   * This method returns the URL of the UMM specification file for this component.
   */
  public String getUmmSpecURL() throws RemoteException
  {
      return ummSpecURL;
  }

  /**
   * This method sets the UMM specificaton url for the component.
   */
  public void setUMMSpecURL(String url) throws RemoteException
  {
    ummSpecURL = url;
    }


    /**
     * This method returns the logic programming access control specifications.
     */
    public String getLogicSpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

            String spec="PROLOG AC SPEC: nothing yet for "+componentName;
            return spec;
    }


    /**
     * This method returns the TLA access control specifications.
     */
    public String getTLASpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

            String spec="TLA SPEC: nothing yet for "+componentName;
            return spec;

    }


    /**
     * This method creates a new student in the student information system
     */

    public boolean addStudent(String sid, SecurityAttributeList sal){

            writeTrace(sid,sal,"ADD","STARTED");

            // Foward this request to the record server.
            try{
                    boolean result = false;
                    result = studentRecordServer.addStudent(sid, sal);
                    if(result)writeTrace(sid,sal,"ADD","SUCCEEDED");
                    if(!result)writeTrace(sid,sal,"ADD","FAILED");
                    return result;

            }catch(Exception e)
        {
            System.out.println(e);
                    return false;
        }
    }

    /**
     * This method reads a student record from the student information system
     */
    public StudentRecord readStudentRecord(String sid, SecurityAttributeList sal) throws
RemoteException{
            writeTrace(sid,sal,"READ","STARTED");
            // Foward this request to the record server.
            StudentRecord sr = studentRecordServer.readStudentRecord(sid,sal );
            writeTrace(sid,sal,"READ","SUCCEEDED");
            return sr;
    }


    /**
     * This method saves student record to the student information system
     */

    public boolean saveStudentRecord(String sid,StudentRecord sr, SecurityAttributeList sal) throws
RemoteException{
            writeTrace(sid,sal,"SAVE","STARTED");
            // Foward this request to the record server.
            try{
                    boolean result=false;
                    result=studentRecordServer.saveStudentRecord(sid,sr,sal);
                    if (result) writeTrace(sid,sal,"SAVE","SUCCEEDED");
                    if (!result) writeTrace(sid,sal,"SAVE","FAILED");
                    return result;

            }catch(Exception e)
        {
            System.out.println(e);
                    return false;
        }
    }


    /**
     * main routine to start the service.
     * Usage: java -Djava.library.path=. UserTerminal [-s IP[:port]/server_name] [-u umm_specification_url]
     */
    public static void main(String[] args)
    {
        if(args.length != 0 && args.length != 2 && args.length != 4)
        {
            System.out.println("Usage: java -Djava.library.path=. UserTerminal [-s IP[:port]/server_name] [-u
umm_specification_url]");
            return;
        }

        String url = null; //host and binding name for the component
        String ummSpecURL = null; //file url for the UMM specification of the component

        if(args.length != 0)
        {
            for(int i = 0; i < args.length; i = i + 2)
            {
                if(args[i].trim().equals("-s"))
                {
                    url = "//" + args[i+1];
                }
                else if(args[i].trim().equals("-u"))
                {
                    ummSpecURL = args[i+1];
                }
            }
        }

        if(url == null)
        {
            url = "//localhost/UserTerminal";
        }

        if(ummSpecURL == null)
        {
            ummSpecURL = "../srs_xml/user_terminal_spec.xml";
        }

        try
        {
            UserTerminal userTerminal = new UserTerminal();
            userTerminal.setUMMSpecURL(ummSpecURL);
            Naming.rebind(url, userTerminal);
            System.out.println("Java UserTerminal is ready:");
            System.out.println("   Host: " + url);
            System.out.println("   UMM Specification URL: " + ummSpecURL);
        }
        catch(RemoteException e)
        {
            System.out.println(e);
        }
        catch(MalformedURLException e)
        {
            System.out.println(e);
        }
    }
}
```

RecordServer2.java

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.net.MalformedURLException;
import java.text.NumberFormat;
import java.io.*;
import java.util.*;

/**
 * This class provides the implementation for the component
 * RecordServer in the univerity domain example. contains guards on gsi
 *
 * @author Alex Crespi
 * @date October 2004
 * @version 1.0
 */
public class RecordServer2 extends UnicastRemoteObject implements IStudentRecord, IConfigure
{
   private IAcademicRecord academicRecordServer;
   private IFinancialAid financialAidServer;
   private IStudentEmployment studentEmploymentServer;
   private ISecurityCenter securityCenter;
   private String academicRecordServer_ID;
   private String financialAidServer_ID;
   private String studentEmploymentServer_ID;
   private String securityCenter_ID;
   private String ummSpecURL;
   private Hashtable gsi;
   private PrintWriter outputFile;

   /**
    * Constructor.
    */
   public RecordServer2() throws RemoteException
   {
      reset_configuration();

         //open output trace files
         try{
                  outputFile=new PrintWriter(new FileWriter(new File("RecordServer2Trace.txt")));
         }catch(Exception e){
                  System.out.println("ERROR:"+e);
         }
   }
   /**
    * This method writes log information about the functioning of this component to a file
    *
    * @param student accessid for the student record being accessed
    * @param sal security attribute for the principal excuting the request
    * @param command the request being made
    * @param comment and information to record. typicall success or failure
    */
   public void writeTrace(String student,SecurityAttributeList sal, String command, String comment){

         String user = "";

         //get user accessid from security attribute list
         for(int k=0; k<sal.size(); k++){
                  SecurityAttribute sa=sal.getAttribute(k);
                  if (sa.attributeType.equals("accessid")){
                           ArrayList value=(ArrayList)sa.value;
                           user=(String)((ArrayList)value).get(0);
                  }
         }
```

```
            Date d=new Date();
            String time = (new Long(d.getTime())).toString();


            try{
                        outputFile.println(time+" ; RecordServer2 ; "+user+" ; "+command+" ; "+ student +" ;
"+comment);
                        outputFile.flush();
            }catch(Exception e){
                        System.out.println("ERROR:"+e);
            }
    }

    /**
     * This method configures a component to work in a bank system.
     *
     * @param server_name IP[:port]/binding_name
     * @param option The values for this parameter is one of TransactionServerManager,
     * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
     */
    public void configure(String server_name, String option) throws RemoteException
    {
       String url;

       if(server_name == null || server_name.trim().equals(""))
          return;

       try
       {
          if(option.equals("AcademicRecordServer"))
          {
             url = "rmi://" + server_name.trim();
             academicRecordServer= (IAcademicRecord)Naming.lookup(url);
             academicRecordServer_ID = server_name.trim();
             System.out.println(option + ":" + server_name);
          }
          else if(option.equals("FinancialAidServer"))
          {
             url = "rmi://" + server_name.trim();
             financialAidServer= (IFinancialAid)Naming.lookup(url);
             financialAidServer_ID = server_name.trim();
             System.out.println(option + ":" + server_name);
          }
          else if(option.equals("StudentEmploymentServer"))
          {
             url = "rmi://" + server_name.trim();
             studentEmploymentServer= (IStudentEmployment)Naming.lookup(url);
             studentEmploymentServer_ID = server_name.trim();
             System.out.println(option + ":" + server_name);
          }
          else if(option.equals("SecurityCenter"))
          {
             url = "rmi://" + server_name.trim();
             securityCenter= (ISecurityCenter)Naming.lookup(url);
             securityCenter_ID = server_name.trim();
             System.out.println(option + ":" + server_name);
          }
       }
       catch(MalformedURLException e)
       {
          System.out.println (e);
       }
       catch(NotBoundException e)
       {
          System.out.println (e);
       }
       catch(ConnectException e)
       {
          System.out.println(e);
       }
       catch(RemoteException e)
       {
          System.out.println (e);
       }
    }

    /**
     * This method deconfigures a component.
     *
     * @param server_name IP[:port]/binding_name
     * @param option The values for this parameter is one of TransactionServerManager,
     * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
     */
    public void deconfigure(String server_name, String option) throws RemoteException
    {

       if(server_name == null || server_name.trim().equals(""))
          return;

       if(option.equals("AcademicRecordServer"))
       {
            academicRecordServer = null;
            academicRecordServer_ID = null;
       }
       else if(option.equals("FinancialAidServer"))
       {
             financialAidServer = null;
             financialAidServer_ID = null;
       }
       else if(option.equals("StudentEmploymentServer"))
       {
             studentEmploymentServer = null;
             studentEmploymentServer_ID = null;
       }
       else if(option.equals("SecurityCenter"))
       {
             securityCenter = null;
             securityCenter_ID = null;
       }
    }

    /**
     * This method returns the configuration of a component.
     *
     * @param option The values for this parameter is one of TransactionServerManager,
     * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
     */
    public Object list_configuration(String option) throws RemoteException
    {
       if(option == null || option.trim().equals(""))
       {
          return null;
       }
       else if(option.equals("AcademicRecordServer"))
       {
          return academicRecordServer_ID;
       }
       else if(option.equals("FinancialAidServer"))
       {
          return financialAidServer_ID;
       }
       else if(option.equals("StudentEmploymentServer"))
       {
          return studentEmploymentServer_ID;
       }
       else if(option.equals("SecurityCenter"))
       {
          return securityCenter_ID;
       }
       else
          return null;
    }

    /**
     * This method resets the configuration for a component.
     */
    public void reset_configuration() throws RemoteException
    {
          gsi =  new Hashtable();

       academicRecordServer = null;
       academicRecordServer_ID = null;

       financialAidServer = null;
       financialAidServer_ID = null;

       studentEmploymentServer = null;
       studentEmploymentServer_ID = null;

       securityCenter = null;
       securityCenter_ID = null;
    }
```

```
   /**
    * This method returns the URL of the UMM specification file for this component.
    */
   public String getUmmSpecURL() throws RemoteException
   {
          return ummSpecURL;
   }


   /**
    * This method sets the UMM specificaton url for the component.
    */
   public void setUMMSpecURL(String url) throws RemoteException
   {
      ummSpecURL = url;
   }


    /**
    * This method returns the logic programming access control specifications.
    */
   public String getLogicSpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

          String spec="PROLOG AC SPEC: nothing yet for "+componentName;
          return spec;
   }


   /**
    * This method returns the TLA access control specifications.
    */
   public String getTLASpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

          String spec="TLA SPEC: nothing yet for "+componentName;
          return spec;
   }

   /**
    * This method creates a new student in the student information system
    */

   public boolean addStudent(String sid, SecurityAttributeList sal) throws RemoteException{

          int ars_tid = 0;
          int fas_tid = 0;
          int ses_tid = 0;

          writeTrace(sid,sal,"ADD","STARTED");

          // Foward this request to the academic record, financial aid, and student employment servers.
          try{
                if (gsi.get(sid)!=null) { return false; }

                ars_tid = academicRecordServer.transaction(sal);
                fas_tid = financialAidServer.transaction(sal);
                ses_tid = studentEmploymentServer.transaction(sal);

                if ( academicRecordServer.addStudent(sid,ars_tid,sal) &&
                    financialAidServer.addStudent(sid,fas_tid,sal) &&
                    studentEmploymentServer.addStudent(sid,ses_tid,sal)  )
          {
                        gsi.put(sid,new GeneralStudentInformation(sid));
                        academicRecordServer.commit(ars_tid,sal);
                        financialAidServer.commit(fas_tid,sal);
                        studentEmploymentServer.commit(ses_tid,sal);
                        writeTrace(sid,sal,"ADD","SUCCEEDED");
                        return true;

                }else{
                        academicRecordServer.rollback(ars_tid,sal);
                        financialAidServer.rollback(fas_tid,sal);
                        studentEmploymentServer.rollback(ses_tid,sal);
                        writeTrace(sid,sal,"ADD","FAILED");
                        return false;
                }

          }catch(Exception e)
      {
          System.out.println(e);
                if(ars_tid!=0) academicRecordServer.rollback(ars_tid,sal);
                if(fas_tid!=0) financialAidServer.rollback(fas_tid,sal);
                if(ses_tid!=0) studentEmploymentServer.rollback(ses_tid,sal);
                writeTrace(sid,sal,"ADD","FAILED");
                return false;
      }
    }

    /**
     * This method reads a student record from the student information system
     */

    public StudentRecord readStudentRecord(String sid, SecurityAttributeList sal) throws
RemoteException{
          writeTrace(sid,sal,"READ","STARTED");

          // Foward this request to the academic record, financial aid, and student employment servers.

          AcademicRecord ar          = academicRecordServer.readAcademicRecord(sid,sal);
          FinancialAid fa            = financialAidServer.readFinancialAid(sid,sal);
          StudentEmployment se       = studentEmploymentServer.readStudentEmployment(sid,sal);

          GeneralStudentInformation gs=null;

          ResourceNameComponentList rncl =new ResourceNameComponentList();
          rncl.addNameComponent(new ResourceNameComponent("sr",sid));
```

```
        rncl.addNameComponent(new ResourceNameComponent("section","gsi"));
        rncl.addNameComponent(new ResourceNameComponent("part","all"));
        ResourceName rn=new ResourceName("UniFrame",rncl);

        System.out.println("PRINCIPAL:"+sal+", RESOURCE:"+rn.toString()+", OPERATION:read");
        if (guard(sal,rn,"read") ){
                gs = (GeneralStudentInformation)gsi.get(sid);
                System.out.println("GUARD ALLOWED ACCESS");
                writeTrace(sid,sal,"READ GSI","ALLOWED");
        }else{
                System.out.println("GUARD BLOCKED ACCESS");
                writeTrace(sid,sal,"READ GSI","BLOCKED");
        }
        return new StudentRecord(sid,gs,ar,fa,se);
    }


    /**
     * This method saves student record to the student information system
    */
    public boolean saveStudentRecord(String sid,StudentRecord sr, SecurityAttributeList sal) throws
RemoteException{
        // Foward this request to the academic record, financial aid, and student employment servers.

        int ars_tid = 0;
        int fas_tid = 0;
        int ses_tid = 0;
        boolean gsi_saved=false;
        boolean ar_saved=false;
        boolean fa_saved=false;
        boolean se_saved=false;

        writeTrace(sid,sal,"SAVE","STARTED");

        try{
                ars_tid = academicRecordServer.transaction(sal);
                fas_tid = financialAidServer.transaction(sal);
                ses_tid = studentEmploymentServer.transaction(sal);

                ResourceNameComponentList rncl =new ResourceNameComponentList();
                rncl.addNameComponent(new ResourceNameComponent("sr",sid));
                rncl.addNameComponent(new ResourceNameComponent("section","gsi"));
                rncl.addNameComponent(new ResourceNameComponent("part","all"));
                ResourceName rn=new ResourceName("UniFrame",rncl);

                System.out.println("PRINCIPAL:"+sal+", RESOURCE:"+rn.toString()+",
OPERATION:save");
                if ( (sr.getGeneralStudentInformation()!=null) && (guard(sal,rn,"save")) ){
                        gsi.put(sid,sr.getGeneralStudentInformation());
                        gsi_saved=true;
                        System.out.println("GUARD ALLOWED ACCESS");
                        writeTrace(sid,sal,"SAVE GSI","ALLOWED");
                }else if (sr.getGeneralStudentInformation()!=null) {
                        System.out.println("GUARD BLOCKED ACCESS");
                        writeTrace(sid,sal,"SAVE GSI","BLOCKED");
                }

                ar_saved = academicRecordServer.saveAcademicRecord(sid,
sr.getAcademicRecord(),ars_tid,sal);
                fa_saved = financialAidServer.saveFinancialAid(sid,sr.getFinancialAid(),fas_tid,sal);
                se_saved = studentEmploymentServer.saveStudentEmployment(sid,
sr.getStudentEmployment(),ses_tid,sal);

                if (sr.getGeneralStudentInformation()==null) gsi_saved=true;
                if (sr.getAcademicRecord()==null) ar_saved=true;
                if (sr.getFinancialAid()==null) fa_saved=true;
                if (sr.getStudentEmployment()==null) se_saved=true;

                if (        gsi_saved && ar_saved && fa_saved && se_saved  )
            {
                        academicRecordServer.commit(ars_tid,sal);
                        financialAidServer.commit(fas_tid,sal);
                        studentEmploymentServer.commit(ses_tid,sal);
                        writeTrace(sid,sal,"SAVE","SUCCEEDED");
                        return true;
                }else{
                        financialAidServer.rollback(fas_tid,sal);
                        academicRecordServer.rollback(ars_tid,sal);
                        studentEmploymentServer.rollback(ses_tid,sal);
                        writeTrace(sid,sal,"SAVE","FAILED");
                        return false;
                }

        }catch(Exception e)
      {
        System.out.println("ERROR:"+e);
                if(ars_tid!=0) academicRecordServer.rollback(ars_tid,sal);
                if(fas_tid!=0) financialAidServer.rollback(fas_tid,sal);
                if(ses_tid!=0) studentEmploymentServer.rollback(ses_tid,sal);
                writeTrace(sid,sal,"SAVE","FAILED");
                return false;
      }
    }

    private boolean guard(SecurityAttributeList sal, ResourceName rn, String operation){
        try{
                SecurityAttributeList new_sal= dynamicAttributes(sal,rn);
                return securityCenter.evaluatePolicy(new_sal,rn,operation);
        }catch(Exception e)
      {
        System.out.println("ERROR:"+e);
                return false;
      }
    }

    private SecurityAttributeList dynamicAttributes(SecurityAttributeList sal,ResourceName rn){
        //if the principal is the student owner of the record, add studentowner to the roles.
        String resourceStudentId=null;
```

```
            SecurityAttribute sa=null;
            String accessId=null;

            ResourceNameComponentList rncl = rn.getNameComponentList();

            //find student id in the student record requested
            for(int i=0; i<rncl.size(); i++){
                        ResourceNameComponent rnc=rncl.getListItem(i);
                        if (rnc.name.equals("sr")) resourceStudentId=rnc.value;
            }


            //find access id in the security attributes
            for(int i=0; i<sal.size(); i++){
                        sa=sal.getAttribute(i);
                        if (sa.attributeType.equals("accessid")){
                                    ArrayList value=(ArrayList)sa.value;
                                    accessId=(String)((ArrayList)value).get(0);
                        }
            }

            //create new secuity attribute list to return
            SecurityAttributeList new_sal=new SecurityAttributeList();

            for(int i=0; i<sal.size(); i++){
                        sa=sal.getAttribute(i);
                        ArrayList attributeValue=new ArrayList();
                        if ( sa.attributeType.equals("roles") || sa.attributeType.equals("groups") ||
sa.attributeType.equals("accessid")){
                                    ArrayList attributeOldValue = (ArrayList)sa.value;
                                    for(int c=0;c<attributeOldValue.size();c++){
                                                attributeValue.add(attributeOldValue.get(c));
                                    }
                                    if ( sa.attributeType.equals("roles") && accessId.equals(resourceStudentId) )
                                                attributeValue.add("owner");
                        }
                        SecurityAttribute new_sa=new
SecurityAttribute(sa.attributeType,sa.definingAuthority,attributeValue);
                        new_sal.addAttribute(new_sa);
            }
            return new_sal;
   }

   /**
    * main routine to start the service.
    * Usage: java -Djava.library.path=. RecordServer [-s IP[:port]/server_name] [-u umm_specification_url]
    */
   public static void main(String[] args)
   {
      if(args.length != 0 && args.length != 2 && args.length != 4)
      {
         System.out.println("Usage: java -Djava.library.path=. RecordServer [-s IP[:port]/server_name] [-u
umm_specification_url]");
         return;
```

```
        }

        String url = null; //host and binding name for the component
        String ummSpecURL = null; //file url for the UMM specification of the component

        if(args.length != 0)
        {
           for(int i = 0; i < args.length; i = i + 2)
           {
              if(args[i].trim().equals("-s"))
              {
                 url = "//" + args[i+1];
              }
              else if(args[i].trim().equals("-u"))
              {
                 ummSpecURL = args[i+1];
              }
           }
        }

        if(url == null)
        {
           url = "//localhost/RecordServer2";
        }

        if(ummSpecURL == null)
        {
           ummSpecURL = "../srs_xml/record_server2_spec.xml";
        }

        try
        {
           RecordServer2 recordServer = new RecordServer2();
           recordServer.setUMMSpecURL(ummSpecURL);
           Naming.rebind(url, recordServer);
           System.out.println("Java RecordServer2 is ready:");
           System.out.println("   Host: " + url);
           System.out.println("   UMM Specification URL: " + ummSpecURL);
        }
        catch(RemoteException e)
        {
           System.out.println(e);
        }
        catch(MalformedURLException e)
        {
           System.out.println(e);
        }
     }
  }
```

AcademicRecordServer2.java

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.net.MalformedURLException;
import java.text.NumberFormat;
import java.io.*;
import java.util.*;

/**
 * This class provides the implementation for the component
 * AcademicRecordServer in the univerity domain example. contains guards on ar
 *
 * @author Alex Crespi
 * @date October 2004
 * @version 1.0
 */
public class AcademicRecordServer2 extends UnicastRemoteObject implements IAcademicRecord,
IConfigure
{
   private ISecurityCenter securityCenter;
   private String securityCenter_ID;
   private String ummSpecURL;
   private Hashtable ar;

   private int transactionCounter=1;
   private Hashtable transactionRecorder;
   private PrintWriter outputFile;

   /**
    * Constructor.
    */
   public AcademicRecordServer2() throws RemoteException
   {
      reset_configuration();

            //open output trace files
            try{
                        outputFile=new PrintWriter(new FileWriter(new
File("AcademicRecordServer2Trace.txt")));
            }catch(Exception e){
                        System.out.println("ERROR:"+e);
            }
   }

   /**
    * This method writes log information about the functioning of this component to a file
    *
    * @param student accessid for the student record being accessed
    * @param sal security attribute for the principal excuting the request
    * @param command the request being made
    * @param comment and information to record. typicall success or failure
    */
```

```
  public void writeTrace(String student,SecurityAttributeList sal, String command, String comment){

        String user = "";

        //get user accessid from security attribute list
        for(int k=0; k<sal.size(); k++){
              SecurityAttribute sa=sal.getAttribute(k);
              if (sa.attributeType.equals("accessid")){
                    ArrayList value=(ArrayList)sa.value;
                    user=(String)((ArrayList)value).get(0);
              }
        }

        Date d=new Date();
        String time = (new Long(d.getTime())).toString();

        try{
              outputFile.println(time+" ; AcademicRecordServer2 ; "+user+" ; "+command+" ; "+
student +" ; "+comment);
              outputFile.flush();
        }catch(Exception e){
              System.out.println("ERROR:"+e);
        }
  }



  /**
   * This method configures a component to work in a bank system.
   *
   * @param server_name IP[:port]/binding_name
   * @param option The values for this parameter is one of TransactionServerManager,
   * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
   */
  public void configure(String server_name, String option) throws RemoteException
  {
    String url;

    if(server_name == null || server_name.trim().equals(""))
      return;

    try
    {
      if(option.equals("SecurityCenter"))
      {
        url = "rmi://" + server_name.trim();
        securityCenter= (ISecurityCenter)Naming.lookup(url);
        securityCenter_ID = server_name.trim();
        System.out.println(option + ":" + server_name);
      }
    }
    catch(MalformedURLException e)
    {
      System.out.println (e);
```

```
        }
        catch(NotBoundException e)
        {
            System.out.println (e);
        }
        catch(ConnectException e)
        {
            System.out.println(e);
        }
        catch(RemoteException e)
        {
            System.out.println (e);
        }
    }

    /**
     * This method deconfigures a component.
     *
     * @param server_name IP[:port]/binding_name
     * @param option The values for this parameter is one of TransactionServerManager,
     * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
     */
    public void deconfigure(String server_name, String option) throws RemoteException
    {

        if(server_name == null || server_name.trim().equals(""))
            return;

        if(option.equals("SecurityCenter"))
        {
                securityCenter = null;
                securityCenter_ID = null;
        }
    }

    /**
     * This method returns the configuration of a component.
     *
     * @param option The values for this parameter is one of TransactionServerManager,
     * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
     */
    public Object list_configuration(String option) throws RemoteException
    {
        if(option.equals("SecurityCenter"))
        {
            return securityCenter_ID;
        }
        else
            return null;
    }

    /**
     * This method resets the configuration for a component.
     */
```

```
   public void reset_configuration() throws RemoteException
   {
         ar =  new Hashtable();

      securityCenter = null;
      securityCenter_ID = null;
         transactionCounter=1;
         transactionRecorder=new Hashtable();
   }

   /**
    * This method returns the URL of the UMM specification file for this component.
    */
   public String getUmmSpecURL() throws RemoteException
   {
         return ummSpecURL;
   }

   /**
    * This method sets the UMM specificaton url for the component.
    */
   public void setUMMSpecURL(String url) throws RemoteException
   {
      ummSpecURL = url;
   }


    /**
    * This method returns the logic programming access control specifications.
    */
   public String getLogicSpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

         String spec="PROLOG AC SPEC: nothing yet for "+componentName;
         return spec;
   }


   /**
    * This method returns the TLA access control specifications.
    */
   public String getTLASpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

         String spec="TLA SPEC: nothing yet for "+componentName;
         return spec;

   }

   /**
    * This method creates a new student in the student information system
    */

   public boolean addStudent(String sid, int tid, SecurityAttributeList sal) throws RemoteException{
```

```
            writeTrace(sid,sal,"ADD","STARTED");
            AcademicRecord arr=new AcademicRecord(sid);
            System.out.println("CREATED:"+arr);
            writeTrace(sid,sal,"ADD","SUCCEEDED");
            ar.put(sid,arr);
            return true;
    }

    /**
     * This method reads a student record from the student information system
     */

    public AcademicRecord readAcademicRecord(String sid, SecurityAttributeList sal) throws
RemoteException{
            writeTrace(sid,sal,"READ AR","STARTED");

            ResourceNameComponentList rncl =new ResourceNameComponentList();
            rncl.addNameComponent(new ResourceNameComponent("sr",sid));
            rncl.addNameComponent(new ResourceNameComponent("section","ar"));
            rncl.addNameComponent(new ResourceNameComponent("part","all"));
            ResourceName rn=new ResourceName("UniFrame",rncl);

            System.out.println("PRINCIPAL:"+sal+", RESOURCE:"+rn.toString()+", OPERATION:read");

            if (guard(sal,rn,"read") ){
                    System.out.println("GUARD ALLOWED ACCESS");
                    writeTrace(sid,sal,"READ AR","ALLOWED");
                    writeTrace(sid,sal,"READ AR","SUCCEEDED");
                    return (AcademicRecord)ar.get(sid);
            }else{
                    System.out.println("GUARD BLOCKED ACCESS");
                    writeTrace(sid,sal,"READ AR","BLOCKED");
                    writeTrace(sid,sal,"READ AR","FAILED");
                    return (AcademicRecord)null;
            }


    }


    /**
     * This method saves student record to the student information system
     */


    public boolean saveAcademicRecord(String sid,AcademicRecord ac, int tid, SecurityAttributeList sal)
throws RemoteException{

            writeTrace(sid,sal,"SAVE AR","STARTED");

            //save previous version of the academic record
            transactionRecorder.put(new Integer(tid),(AcademicRecord)ar.get(sid));

            if (ac!=null) {
```

```
                        ResourceNameComponentList rncl =new ResourceNameComponentList();
                        rncl.addNameComponent(new ResourceNameComponent("sr",sid));
                        rncl.addNameComponent(new ResourceNameComponent("section","ar"));
                        rncl.addNameComponent(new ResourceNameComponent("part","all"));
                        ResourceName rn=new ResourceName("UniFrame",rncl);

                        System.out.println("PRINCIPAL:"+sal+", RESOURCE:"+rn.toString()+",
OPERATION:save");

                        if (guard(sal,rn,"save") ){
                                System.out.println("GUARD ALLOWED ACCESS");
                                ar.put(sid,ac);
                                writeTrace(sid,sal,"SAVE AR","ALLOWED");
                                writeTrace(sid,sal,"SAVE AR","SUCCEEDED");
                                return true;
                        }else{
                                System.out.println("GUARD BLOCKED ACCESS");
                                writeTrace(sid,sal,"SAVE AR","BLOCKED");
                                writeTrace(sid,sal,"SAVE AR","FAILED");
                                return false;
                        }
                }

                return false;


    }



    public int transaction( SecurityAttributeList sal){
            writeTrace("N/A",sal,"TRANSACTION","STARTED");
            return transactionCounter++;
    }

    public void commit(int tid, SecurityAttributeList sal) throws RemoteException{
            writeTrace("N/A",sal,"TRANSACTION","COMMITTED");
            transactionRecorder.remove(new Integer(tid));
    }

    public void rollback(int tid, SecurityAttributeList sal) throws RemoteException{
            AcademicRecord savedRecord = (AcademicRecord)transactionRecorder.get(new Integer(tid));
            writeTrace("N/A",sal,"TRANSACTION","ROLLEDBACK");
            if(savedRecord!=null) ar.put(savedRecord.getStudentId(),savedRecord);
    }


    private boolean guard(SecurityAttributeList sal, ResourceName rn, String operation){
            try{
                    SecurityAttributeList new_sal= dynamicAttributes(sal,rn);
                    return securityCenter.evaluatePolicy(new_sal,rn,operation);
            }catch(Exception e)
      {
            System.out.println("ERROR:"+e);
```

```
                    return false;
    }

   }

   private SecurityAttributeList dynamicAttributes(SecurityAttributeList sal,ResourceName rn){

          //if the principal is the student owner of the record, add studentowner to the roles.
          String resourceStudentId=null;
          SecurityAttribute sa=null;
          String accessId=null;

          ResourceNameComponentList rncl = rn.getNameComponentList();

          //find student id in the student record requested
          for(int i=0; i<rncl.size(); i++){
                    ResourceNameComponent rnc=rncl.getListItem(i);
                    if (rnc.name.equals("sr")) resourceStudentId=rnc.value;
          }


          //find access id in the security attributes
          for(int i=0; i<sal.size(); i++){
                    sa=sal.getAttribute(i);
                    if (sa.attributeType.equals("accessid")){
                              ArrayList value=(ArrayList)sa.value;
                              accessId=(String)((ArrayList)value).get(0);
                    }
          }

          //create new secuity attribute list to return
          SecurityAttributeList new_sal=new SecurityAttributeList();

          for(int i=0; i<sal.size(); i++){
                    sa=sal.getAttribute(i);
                    ArrayList attributeValue=new ArrayList();
                    if ( sa.attributeType.equals("roles") || sa.attributeType.equals("groups") ||
sa.attributeType.equals("accessid")){
                              ArrayList attributeOldValue = (ArrayList)sa.value;
                              for(int c=0;c<attributeOldValue.size();c++){
                                        attributeValue.add(attributeOldValue.get(c));
                              }
                              if ( sa.attributeType.equals("roles") && accessId.equals(resourceStudentId) )
                                        attributeValue.add("owner");
                    }

                    SecurityAttribute new_sa=new
SecurityAttribute(sa.attributeType,sa.definingAuthority,attributeValue);
                    new_sal.addAttribute(new_sa);
          }

          return new_sal;
   }
```

```
    /**
     * main routine to start the service.
     * Usage: java -Djava.library.path=. AcademicRecordServer [-s IP[:port]/server_name]
     * [-umm_specification_url]
     */

    public static void main(String[] args)
    {
      if(args.length != 0 && args.length != 2 && args.length != 4)
      {
        System.out.println("Usage: java -Djava.library.path=. AcademicRecordServer [-s
IP[:port]/server_name] [-u umm_specification_url]");
        return;
      }

      String url = null; //host and binding name for the component
      String ummSpecURL = null; //file url for the UMM specification of the component

      if(args.length != 0)
      {
        for(int i = 0; i < args.length; i = i + 2)
        {
          if(args[i].trim().equals("-s"))
          {
            url = "//" + args[i+1];
          }
          else if(args[i].trim().equals("-u"))
          {
            ummSpecURL = args[i+1];
          }
        }
      }

      if(url == null)
      {
        url = "//localhost/AcademicRecordServer2";
      }

      if(ummSpecURL == null)
      {
        ummSpecURL = "../srs_xml/academic_record_server2_spec.xml";
      }

      try
      {
        AcademicRecordServer2 academicRecordServer = new AcademicRecordServer2();
        academicRecordServer.setUMMSpecURL(ummSpecURL);
        Naming.rebind(url, academicRecordServer);
        System.out.println("Java AcademicRecordServer2 is ready:");
        System.out.println("   Host: " + url);
        System.out.println("   UMM Specification URL: " + ummSpecURL);
      }
      catch(RemoteException e)
```

```
        {
            System.out.println(e);
        }
        catch(MalformedURLException e)
        {
            System.out.println(e);
        }
    }
}
```

FinancialAidServer2.java

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.net.MalformedURLException;
import java.text.NumberFormat;
import java.io.*;
import java.util.*;

/**
 * This class provides the implementation for the component
 * FinancialServer in the univerity domain example. contains guards on fa
 *
 * @author Alex Crespi
 * @date October 2004
 * @version 1.0
 */
public class FinancialAidServer2 extends UnicastRemoteObject implements IFinancialAid, IConfigure
{
    private ISecurityCenter securityCenter;
    private String securityCenter_ID;
    private String ummSpecURL;
    private Hashtable fa;

    private int transactionCounter=1;
    private Hashtable transactionRecorder;
    private PrintWriter outputFile;

    /**
     * Constructor.
     */
    public FinancialAidServer2() throws RemoteException
    {
     reset_configuration();
            //open output trace files
            try{
                    outputFile=new PrintWriter(new FileWriter(new File("FinancialAidServer2Trace.txt")));
            }catch(Exception e){
                    System.out.println("ERROR:"+e);
            }
    }
    /**
     * This method writes log information about the functioning of this component to a file
     *
     * @param student accessid for the student record being accessed
     * @param sal security attribute for the principal excuting the request
     * @param command the request being made
     * @param comment and information to record. typicall success or failure
     */

    public void writeTrace(String student,SecurityAttributeList sal, String command, String comment){

            String user = "";

            //get user accessid from security attribute list
            for(int k=0; k<sal.size(); k++){
                    SecurityAttribute sa=sal.getAttribute(k);
                    if (sa.attributeType.equals("accessid")){
                            ArrayList value=(ArrayList)sa.value;
                            user=(String)((ArrayList)value).get(0);
                    }
            }

            Date d=new Date();
            String time = (new Long(d.getTime())).toString();

            try{
                    outputFile.println(time+" ; FinancialAidServer2 ; "+user+" ; "+command+" ; "+ student
+" ; "+comment);
                    outputFile.flush();
            }catch(Exception e){
                    System.out.println("ERROR:"+e);
            }
    }



    /**
     * This method configures a component to work in a student information system system.
     *
     * @param server_name IP[:port]/binding_name
     * @param option The values for this parameter is one of SecurityCenter,
     * RecordServer, StudentEmploymentServer, AcademicRecordServer, or FinancialAid
     */
    public void configure(String server_name, String option) throws RemoteException
    {
      String url;

      if(server_name == null || server_name.trim().equals(""))
        return;

      try
      {
        if(option.equals("SecurityCenter"))
        {
              url = "rmi://" + server_name.trim();
              securityCenter= (ISecurityCenter)Naming.lookup(url);
              securityCenter_ID = server_name.trim();
              System.out.println(option + ":" + server_name);
        }
      }
      catch(MalformedURLException e)
      {
        System.out.println (e);
      }
      catch(NotBoundException e)
      {
        System.out.println (e);
      }
      catch(ConnectException e)
      {
        System.out.println(e);
      }
      catch(RemoteException e)
      {
        System.out.println (e);
      }
    }

    /**
     * This method deconfigures a component.
     *
     * @param server_name IP[:port]/binding_name
     * @param option The values for this parameter is one of SecurityCenter,
     * RecordServer, StudentEmploymentServer, AcademicRecordServer, or FinancialAid
     */
    public void deconfigure(String server_name, String option) throws RemoteException
    {

      if(server_name == null || server_name.trim().equals(""))
        return;

      if(option.equals("SecurityCenter"))
      {
              securityCenter = null;
              securityCenter_ID = null;
      }
    }

    /**
     * This method returns the configuration of a component.
     *
     * @param option The values for this parameter is one of SecurityCenter,
     * RecordServer, StudentEmploymentServer, AcademicRecordServer, or FinancialAid
     */
    public Object list_configuration(String option) throws RemoteException
    {
      if(option.equals("SecurityCenter"))
      {
        return securityCenter_ID;
      }
      else
        return null;
    }

    /**
     * This method resets the configuration for a component.
     */
    public void reset_configuration() throws RemoteException
    {
          fa =  new Hashtable();

      securityCenter = null;
      securityCenter_ID = null;
          transactionCounter=1;
          transactionRecorder=new Hashtable();
    }

    /**
     * This method returns the URL of the UMM specification file for this component.
     */
    public String getUmmSpecURL() throws RemoteException
    {
          return ummSpecURL;
    }

    /**
     * This method sets the UMM specificaton url for the component.
     */
    public void setUMMSpecURL(String url) throws RemoteException
    {
      ummSpecURL = url;
    }


     /**
     * This method returns the logic programming access control specifications.
     */
    public String getLogicSpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

          String spec="PROLOG AC SPEC: nothing yet for "+componentName;
          return spec;
    }


    /**
     * This method returns the TLA access control specifications.
     */
    public String getTLASpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

          String spec="TLA SPEC: nothing yet for "+componentName;
        return spec;

    }

    /**
     * This method creates a new student in the student information system
     */
    public boolean addStudent(String sid, int tid, SecurityAttributeList sal) throws RemoteException{
        writeTrace(sid,sal,"ADD","STARTED");
        FinancialAid far=new FinancialAid(sid);
        System.out.println("CREATED:"+far);
        writeTrace(sid,sal,"ADD","SUCCEEDED");
        fa.put(sid,far);
        return true;
    }

    /**
     * This method reads a student record from the student information system
     */
    public FinancialAid readFinancialAid(String sid, SecurityAttributeList sal) throws RemoteException{
        writeTrace(sid,sal,"READ FA","STARTED");

        ResourceNameComponentList rncl =new ResourceNameComponentList();
        rncl.addNameComponent(new ResourceNameComponent("sr",sid));
        rncl.addNameComponent(new ResourceNameComponent("section","fa"));
        rncl.addNameComponent(new ResourceNameComponent("part","all"));
        ResourceName rn=new ResourceName("UniFrame",rncl);

        System.out.println("PRINCIPAL:"+sal+", RESOURCE:"+rn.toString()+", OPERATION:read");

        if (guard(sal,rn,"read") ){
            System.out.println("GUARD ALLOWED ACCESS");
            writeTrace(sid,sal,"READ FA","ALLOWED");
            writeTrace(sid,sal,"READ FA","SUCCEEDED");
            return (FinancialAid)fa.get(sid);
        }else{
            System.out.println("GUARD BLOCKED ACCESS");
            writeTrace(sid,sal,"READ FA","BLOCKED");
            writeTrace(sid,sal,"READ FA","FAILED");
            return (FinancialAid)null;
        }


    }


    /**
     * This method saves student record to the student information system
     */
    public boolean saveFinancialAid(String sid,FinancialAid fi, int tid, SecurityAttributeList sal) throws
RemoteException{
        writeTrace(sid,sal,"SAVE FA","STARTED");

        //save previous version of the academic record
        transactionRecorder.put(new Integer(tid),(FinancialAid)fa.get(sid));

        if (fi!=null) {
            ResourceNameComponentList rncl =new ResourceNameComponentList();
            rncl.addNameComponent(new ResourceNameComponent("sr",sid));
            rncl.addNameComponent(new ResourceNameComponent("section","fa"));
            rncl.addNameComponent(new ResourceNameComponent("part","all"));
            ResourceName rn=new ResourceName("UniFrame",rncl);

            System.out.println("PRINCIPAL:"+sal+", RESOURCE:"+rn.toString()+",
OPERATION:save");

            if (guard(sal,rn,"save") ){
                    System.out.println("GUARD ALLOWED ACCESS");
                    fa.put(sid,fi);
                    writeTrace(sid,sal,"SAVE FA","ALLOWED");
                    writeTrace(sid,sal,"SAVE FA","SUCCEEDED");
                    return true;
            }else{
                    System.out.println("GUARD BLOCKED ACCESS");
                    writeTrace(sid,sal,"SAVE FA","BLOCKED");
                    writeTrace(sid,sal,"SAVE FA","FAILED");
                    return false;
            }

        }

        return false;
    }


    /**
     * This method starts a transaction.
     *
     * @param sal is a security attribute list for the principal
     */
    public int transaction( SecurityAttributeList sal){
        writeTrace("N/A",sal,"TRANSACTION","STARTED");
        return transactionCounter++;
    }

    /**
     * This method commits a transaction.
     *
     * @param sal is a security attribute list for the principal
     */
    public void commit(int tid, SecurityAttributeList sal) throws RemoteException{
        writeTrace("N/A",sal,"TRANSACTION","COMMITTED");
        transactionRecorder.remove(new Integer(tid));
    }

    /* This method rollsback a transaction.
     *
     * @param sal is a security attribute list for the principal
```

```
 */
public void rollback(int tid, SecurityAttributeList sal) throws RemoteException{
        FinancialAid savedRecord = (FinancialAid)transactionRecorder.get(new Integer(tid));
        writeTrace("N/A",sal,"TRANSACTION","ROLLEDBACK");
        if(savedRecord!=null) fa.put(savedRecord.getStudentId(),savedRecord);
}



/* This method implements the access control guard for this component.
 *
 * @param sal is a security attribute list for the principal
 * @param rn is a resource name
 * @param operation is the operation to be performed
 */
private boolean guard(SecurityAttributeList sal, ResourceName rn, String operation){
        try{
                SecurityAttributeList new_sal= dynamicAttributes(sal,rn);
                return securityCenter.evaluatePolicy(new_sal,rn,operation);
        }catch(Exception e)
  {
        System.out.println("ERROR:"+e);
                return false;
  }

}

/* This method modifies a security attribute list to include dynamic attributes.
 *
 * @param sal is a security attribute list for the principal
 * @param rn is a resource name
 */
private SecurityAttributeList dynamicAttributes(SecurityAttributeList sal,ResourceName rn){

        //if the principal is the student owner of the record, add studentowner to the roles.
        String resourceStudentId=null;
        SecurityAttribute sa=null;
        String accessId=null;

        ResourceNameComponentList rncl = rn.getNameComponentList();

        //find student id in the student record requested
        for(int i=0; i<rncl.size(); i++){
                ResourceNameComponent rnc=rncl.getListItem(i);
                if (rnc.name.equals("sr")) resourceStudentId=rnc.value;
        }


        //find access id in the security attributes
        for(int i=0; i<sal.size(); i++){
                sa=sal.getAttribute(i);
                if (sa.attributeType.equals("accessid")){
                        ArrayList value=(ArrayList)sa.value;
                        accessId=(String)((ArrayList)value).get(0);
                        }
                }

                //create new secuity attribute list to return
                SecurityAttributeList new_sal=new SecurityAttributeList();

                for(int i=0; i<sal.size(); i++){
                        sa=sal.getAttribute(i);
                        ArrayList attributeValue=new ArrayList();
                        if ( sa.attributeType.equals("roles") || sa.attributeType.equals("groups") ||
sa.attributeType.equals("accessid")){
                                ArrayList attributeOldValue = (ArrayList)sa.value;
                                for(int c=0;c<attributeOldValue.size();c++){
                                        attributeValue.add(attributeOldValue.get(c));
                                }
                                if ( sa.attributeType.equals("roles") && accessId.equals(resourceStudentId) )
                                        attributeValue.add("owner");
                        }

                        SecurityAttribute new_sa=new
SecurityAttribute(sa.attributeType,sa.definingAuthority,attributeValue);
                        new_sal.addAttribute(new_sa);
                }

                return new_sal;
    }



    /**
     * main routine to start the service.
     * Usage: java -Djava.library.path=. FinancialAidServer [-s IP[:port]/server_name] [-u
umm_specification_url]
     */
    public static void main(String[] args)
    {
        if(args.length != 0 && args.length != 2 && args.length != 4)
        {
            System.out.println("Usage: java -Djava.library.path=. FinancialAidServer [-s
IP[:port]/server_name] [-u umm_specification_url]");
            return;
        }

        String url = null; //host and binding name for the component
        String ummSpecURL = null; //file url for the UMM specification of the component

        if(args.length != 0)
        {
            for(int i = 0; i < args.length; i = i + 2)
            {
                if(args[i].trim().equals("-s"))
                {
                    url = "//" + args[i+1];
                }
            else if(args[i].trim().equals("-u"))
            {
               ummSpecURL = args[i+1];
            }
         }
      }

      if(url == null)
      {
         url = "//localhost/FinancialAidServer2";
      }

      if(ummSpecURL == null)
      {
         ummSpecURL = "../srs_xml/financial_aid_server2_spec.xml";
      }

      try
      {
         FinancialAidServer2 financialAidServer = new FinancialAidServer2();
         financialAidServer.setUMMSpecURL(ummSpecURL);
         Naming.rebind(url, financialAidServer);
         System.out.println("Java FinancialAidServer2 is ready:");
         System.out.println("   Host: " + url);
         System.out.println("   UMM Specification URL: " + ummSpecURL);
      }
      catch(RemoteException e)
      {
         System.out.println(e);
      }
      catch(MalformedURLException e)
      {
         System.out.println(e);
      }
   }
}
```

StudentEmploymentServer2.java

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.net.MalformedURLException;
import java.text.NumberFormat;
import java.io.*;
import java.util.*;

/**
 * This class provides the implementation for the component
 * StudentEmploymentServer in the univerity domain example. contains guards on se
 *
 * @author Alex Crespi
 * @date October 2004
 * @version 1.0
 */
public class StudentEmploymentServer2 extends UnicastRemoteObject implements IStudentEmployment,
IConfigure
{
   private ISecurityCenter securityCenter;
   private String securityCenter_ID;
   private String ummSpecURL;
   private Hashtable se;

   private int transactionCounter=1;
   private Hashtable transactionRecorder;
   private PrintWriter outputFile;

   /**
    * Constructor.
    */
   public StudentEmploymentServer2() throws RemoteException
   {
      reset_configuration();

            //open output trace files
            try{
                        outputFile=new PrintWriter(new FileWriter(new
File("StudentEmploymentServer2Trace.txt")));
            }catch(Exception e){
                        System.out.println("ERROR:"+e);
            }
   }

   /**
    * This method writes log information about the functioning of this component to a file
    *
    * @param student accessid for the student record being accessed
    * @param sal security attribute for the principal excuting the request
    * @param command the request being made
    * @param comment and information to record. typicall success or failure
    */
   public void writeTrace(String student,SecurityAttributeList sal, String command, String comment){

            String user = "";

            //get user accessid from security attribute list
            for(int k=0; k<sal.size(); k++){
                        SecurityAttribute sa=sal.getAttribute(k);
                        if (sa.attributeType.equals("accessid")){
                                    ArrayList value=(ArrayList)sa.value;
                                    user=(String)((ArrayList)value).get(0);
                        }
            }

            Date d=new Date();
            String time = (new Long(d.getTime())).toString();
```

```
        try{
                outputFile.println(time+" ; StudentEmploymentServer2 ; "+user+" ; "+command+" ; "+
student +" ; "+comment);
                outputFile.flush();
        }catch(Exception e){
                System.out.println("ERROR:"+e);
        }
  }




  /**
   * This method configures a component to work in a bank system.
   *
   * @param server_name IP[:port]/binding_name
   * @param option The values for this parameter is one of TransactionServerManager,
   * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
   */
  public void configure(String server_name, String option) throws RemoteException
  {
    String url;

    if(server_name == null || server_name.trim().equals(""))
      return;

    try
    {
      if(option.equals("SecurityCenter"))
      {
        url = "rmi://" + server_name.trim();
        securityCenter= (ISecurityCenter)Naming.lookup(url);
        securityCenter_ID = server_name.trim();
        System.out.println(option + ":" + server_name);
      }
    }
    catch(MalformedURLException e)
    {
      System.out.println (e);
    }
    catch(NotBoundException e)
    {
      System.out.println (e);
    }
    catch(ConnectException e)
    {
      System.out.println(e);
    }
    catch(RemoteException e)
    {
      System.out.println (e);
    }
  }
```

```
/**
 * This method deconfigures a component.
 *
 * @param server_name IP[:port]/binding_name
 * @param option The values for this parameter is one of TransactionServerManager,
 * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
 */
public void deconfigure(String server_name, String option) throws RemoteException
{

   if(server_name == null || server_name.trim().equals(""))
      return;

   if(option.equals("SecurityCenter"))
   {
         securityCenter = null;
         securityCenter_ID = null;
   }
}

/**
 * This method returns the configuration of a component.
 *
 * @param option The values for this parameter is one of TransactionServerManager,
 * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
 */
public Object list_configuration(String option) throws RemoteException
{
   if(option.equals("SecurityCenter"))
   {
      return securityCenter_ID;
   }
   else
      return null;
}

/**
 * This method resets the configuration for a component.
 */
public void reset_configuration() throws RemoteException
{
        se =  new Hashtable();

   securityCenter = null;
   securityCenter_ID = null;
        transactionCounter=1;
        transactionRecorder=new Hashtable();
}

/**
 * This method returns the URL of the UMM specification file for this component.
 */
public String getUmmSpecURL() throws RemoteException
{
```

```
            return ummSpecURL;
    }

    /**
     * This method sets the UMM specificaton url for the component.
     */
    public void setUMMSpecURL(String url) throws RemoteException
    {
        ummSpecURL = url;
    }


     /**
     * This method returns the logic programming access control specifications.
     */
    public String getLogicSpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

            String spec="PROLOG AC SPEC: nothing yet for "+componentName;
            return spec;
    }


    /**
     * This method returns the TLA access control specifications.
     */
    public String getTLASpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException{

            String spec="TLA SPEC: nothing yet for "+componentName;
            return spec;

    }

    /**
     * This method creates a new student in the student information system
     */

    public boolean addStudent(String sid, int tid, SecurityAttributeList sal) throws RemoteException{
            writeTrace(sid,sal,"ADD","STARTED");
            StudentEmployment ser=new StudentEmployment(sid);
            System.out.println("CREATED:"+ser);
            writeTrace(sid,sal,"ADD","SUCCEEDED");
            se.put(sid,ser);
            return true;
    }

    /**
     * This method reads a student record from the student information system
     */

    public StudentEmployment readStudentEmployment(String sid, SecurityAttributeList sal) throws
RemoteException{
            writeTrace(sid,sal,"READ SE","STARTED");
            ResourceNameComponentList rncl =new ResourceNameComponentList();
            rncl.addNameComponent(new ResourceNameComponent("sr",sid));
            rncl.addNameComponent(new ResourceNameComponent("section","se"));
            rncl.addNameComponent(new ResourceNameComponent("part","all"));
            ResourceName rn=new ResourceName("UniFrame",rncl);


            System.out.println("PRINCIPAL:"+sal+", RESOURCE:"+rn.toString()+", OPERATION:read");

            if (guard(sal,rn,"read") ){
                    System.out.println("GUARD ALLOWED ACCESS");
                    writeTrace(sid,sal,"READ SE","ALLOWED");
                    writeTrace(sid,sal,"READ SE","SUCCEEDED");
                    return (StudentEmployment)se.get(sid);
            }else{
                    System.out.println("GUARD BLOCKED ACCESS");
                    writeTrace(sid,sal,"READ SE","BLOCKED");
                    writeTrace(sid,sal,"READ SE","FAILED");
                    return (StudentEmployment)null;
            }


    }


    /**
     * This method saves student record to the student information system
     */


    public boolean saveStudentEmployment(String sid,StudentEmployment st, int tid, SecurityAttributeList
sal) throws RemoteException{
            writeTrace(sid,sal,"SAVE SE","STARTED");

            //save previous version of the academic record
            transactionRecorder.put(new Integer(tid),(StudentEmployment)se.get(sid));

            if (st!=null) {
                    ResourceNameComponentList rncl =new ResourceNameComponentList();
                    rncl.addNameComponent(new ResourceNameComponent("sr",sid));
                    rncl.addNameComponent(new ResourceNameComponent("section","se"));
                    rncl.addNameComponent(new ResourceNameComponent("part","all"));
                    ResourceName rn=new ResourceName("UniFrame",rncl);

                    System.out.println("PRINCIPAL:"+sal+", RESOURCE:"+rn.toString()+",
OPERATION:save");

                    if (guard(sal,rn,"save") ){
                            System.out.println("GUARD ALLOWED ACCESS");
                            se.put(sid,st);
                            writeTrace(sid,sal,"SAVE SE","ALLOWED");
                            writeTrace(sid,sal,"SAVE SE","SUCCEEDED");
                            return true;
                    }else{
                            System.out.println("GUARD BLOCKED ACCESS");
                            writeTrace(sid,sal,"SAVE SE","BLOCKED");
                            writeTrace(sid,sal,"SAVE SE","FAILED");
                            return false;
                    }
            }

            return false;
    }



    public int transaction(SecurityAttributeList sal){
            writeTrace("N/A",sal,"TRANSACTION","STARTED");
            return transactionCounter++;
    }

    public void commit(int tid, SecurityAttributeList sal) throws RemoteException{
            writeTrace("N/A",sal,"TRANSACTION","COMMITTED");
            transactionRecorder.remove(new Integer(tid));
    }

    public void rollback(int tid, SecurityAttributeList sal) throws RemoteException{
            StudentEmployment savedRecord = (StudentEmployment)transactionRecorder.get(new
Integer(tid));
            writeTrace("N/A",sal,"TRANSACTION","ROLLEDBACK");
            if(savedRecord!=null) se.put(savedRecord.getStudentId(),savedRecord);
    }



    private boolean guard(SecurityAttributeList sal, ResourceName rn, String operation){
            try{
                    SecurityAttributeList new_sal= dynamicAttributes(sal,rn);
                    return securityCenter.evaluatePolicy(new_sal,rn,operation);
            }catch(Exception e)
         {
            System.out.println("ERROR:"+e);
                    return false;
         }

    }

    private SecurityAttributeList dynamicAttributes(SecurityAttributeList sal,ResourceName rn){

            //if the principal is the student owner of the record, add studentowner to the roles.
            String resourceStudentId=null;
            SecurityAttribute sa=null;
            String accessId=null;

            ResourceNameComponentList rncl = rn.getNameComponentList();

            //find student id in the student record requested
            for(int i=0; i<rncl.size(); i++){
                    ResourceNameComponent rnc=rncl.getListItem(i);
                    if (rnc.name.equals("sr")) resourceStudentId=rnc.value;
            }


            //find access id in the security attributes
            for(int i=0; i<sal.size(); i++){
                    sa=sal.getAttribute(i);
                    if (sa.attributeType.equals("accessid")){
                            ArrayList value=(ArrayList)sa.value;
                            accessId=(String)((ArrayList)value).get(0);
                    }
            }

            //create new secuity attribute list to return
            SecurityAttributeList new_sal=new SecurityAttributeList();

            for(int i=0; i<sal.size(); i++){
                    sa=sal.getAttribute(i);
                    ArrayList attributeValue=new ArrayList();
                    if ( sa.attributeType.equals("roles") || sa.attributeType.equals("groups") ||
sa.attributeType.equals("accessid")){
                            ArrayList attributeOldValue = (ArrayList)sa.value;
                            for(int c=0;c<attributeOldValue.size();c++){
                                    attributeValue.add(attributeOldValue.get(c));
                            }
                            if ( sa.attributeType.equals("roles") && accessId.equals(resourceStudentId) )
                                    attributeValue.add("owner");
                    }

                    SecurityAttribute new_sa=new
SecurityAttribute(sa.attributeType,sa.definingAuthority,attributeValue);
                    new_sal.addAttribute(new_sa);
            }

            return new_sal;
    }


    /**
     * main routine to start the service.
     * Usage: java -Djava.library.path=. StudentEmploymentServer [-s IP[:port]/server_name] [-u
umm_specification_url]
     */
    public static void main(String[] args)
    {
      if(args.length != 0 && args.length != 2 && args.length != 4)
      {
        System.out.println("Usage: java -Djava.library.path=. StudentEmploymentServer[-s
IP[:port]/server_name] [-u umm_specification_url]");
        return;
      }

      String url = null; //host and binding name for the component
```

```
        String ummSpecURL = null; //file url for the UMM specification of the component

        if(args.length != 0)
        {
           for(int i = 0; i < args.length; i = i + 2)
           {
              if(args[i].trim().equals("-s"))
              {
                 url = "//" + args[i+1];
              }
              else if(args[i].trim().equals("-u"))
              {
                 ummSpecURL = args[i+1];
              }
           }
        }

        if(url == null)
        {
           url = "//localhost/StudentEmploymentServer2";
        }

        if(ummSpecURL == null)
        {
           ummSpecURL = "../srs_xml/student_employment_server2_spec.xml";
        }

        try
        {
           StudentEmploymentServer2 studentEmploymentServer = new StudentEmploymentServer2();
           studentEmploymentServer .setUMMSpecURL(ummSpecURL);
           Naming.rebind(url, studentEmploymentServer);
           System.out.println("Java StudentEmploymentServer2 is ready:");
           System.out.println("   Host: " + url);
           System.out.println("   UMM Specification URL: " + ummSpecURL);
        }
        catch(RemoteException e)
        {
           System.out.println(e);
        }
        catch(MalformedURLException e)
        {
           System.out.println(e);
        }
    }
}
```

SecurityCenter.java

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.net.MalformedURLException;
import java.text.NumberFormat;
import java.io.*;
import java.util.*;

/**
 * This class provides the implementation for the component
 * SecurityCenter in the univerity domain example.
 *
 * @author Alex Crespi
 * @date October 2004
 * @version 1.0
 */
public class SecurityCenter extends UnicastRemoteObject implements ISecurityCenter
{
   private String ummSpecURL;


   /**
    * Constructor.
    */
   public SecurityCenter() throws RemoteException
   {
      reset_configuration();
   }

   /**
    * This method configures a component to work in a bank system.
    *
    * @param server_name IP[:port]/binding_name
    * @param option The values for this parameter is one of TransactionServerManager,
    * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
    */
   public void configure(String server_name, String option) throws RemoteException
   {
   }

   /**
    * This method deconfigures a component.
    *
    * @param server_name IP[:port]/binding_name
    * @param option The values for this parameter is one of TransactionServerManager,
    * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
    */
   public void deconfigure(String server_name, String option) throws RemoteException
   {

   }

   /**
    * This method returns the configuration of a component.
    *
    * @param option The values for this parameter is one of TransactionServerManager,
    * TransactionServer, DeluxeTransactionServer, EconomicTransactionServer, CustomerValidationServer
```

```
     */
    public Object list_configuration(String option) throws RemoteException
    {
            return new Object();
    }

    /**
     * This method resets the configuration for a component.
     */
    public void reset_configuration() throws RemoteException
    {
    }

    /**
     * This method returns the URL of the UMM specification file for this component.
     */
    public String getUmmSpecURL() throws RemoteException
    {
            return ummSpecURL;
    }

    /**
     * This method sets the UMM specificaton url for the component.
     */
    public void setUMMSpecURL(String url) throws RemoteException
    {
       ummSpecURL = url;
    }


    public boolean evaluatePolicy(SecurityAttributeList sal, ResourceName resource, String operation)
throws RemoteException
    {
            System.out.println("ACCESS CONTROL REQUEST:");
            System.out.println("   Security Attributes:");
            System.out.println("         "+sal);

            System.out.println("   Resource Name:");
            System.out.println("         "+resource);
            System.out.println("   Operation:");
            System.out.println("         "+operation);

            boolean result=false;

            String  accessId="";
            boolean owner=false;
            boolean student=false;
            boolean staff=false;
            boolean academicadvisor=false;
            boolean bursar=false;

            String resourceStudentId="";
            String section="";
            String part="";
```

```
ResourceNameComponentList rncl = resource.getNameComponentList();
SecurityAttribute sa=null;

//find portions of the resource name
for(int i=0; i<rncl.size(); i++){
        ResourceNameComponent rnc=rncl.getListItem(i);
        if (rnc.name.equals("sr")) resourceStudentId=rnc.value;
        if (rnc.name.equals("section")) section=rnc.value;
        if (rnc.name.equals("part")) part=rnc.value;
}

//find portions of the security attributes
for(int i=0; i<sal.size(); i++){
        sa=sal.getAttribute(i);
        if (sa.attributeType.equals("accessid")){
                ArrayList value=(ArrayList)sa.value;
                accessId=(String)((ArrayList)value).get(0);
        }

        if (sa.attributeType.equals("roles")){
                ArrayList value=(ArrayList)sa.value;
                for(int c=0;c<value.size();c++){
                        if(((String)((ArrayList)value).get(c)).equals("owner")) owner=true;
                        if(((String)((ArrayList)value).get(c)).equals("student")) student=true;
                        if(((String)((ArrayList)value).get(c)).equals("academicadvisor"))
academicadvisor=true;
                        if(((String)((ArrayList)value).get(c)).equals("bursar")) bursar=true;
                }
        }

        if (sa.attributeType.equals("groups")){
                ArrayList value=(ArrayList)sa.value;
                for(int c=0;c<value.size();c++){
                        if(((String)((ArrayList)value).get(c)).equals("staff")) staff=true;
                }
        }

}

System.out.println("accessid "+accessId);
System.out.println("owner "+owner);
System.out.println("student "+student);
System.out.println("staff "+staff);
System.out.println("academicadvisor "+academicadvisor);
System.out.println("bursar "+bursar);
System.out.println("section "+section);
System.out.println("part "+part);

//if the section is gsi and owner is in the roles and operation is save, then allow
if (section.equals("gsi") && owner && operation.equals("save")) result=true;

//if the section is anything and owner is in the roles and operation is read, then allow
if ( owner && operation.equals("read") )result=true;
```

```
            //if the section is gsi and staff in groups and operation is read, then allow
            if (section.equals("gsi") && staff && operation.equals("read")) result=true;


            //if section is ar and academicadvisor is in role and operation is read or save, then allow
            if (section.equals("ar") && academicadvisor && (operation.equals("read") ||
operation.equals("save") )) result=true;


            //if section is fa or se and bursar is in role and operation is read or save, then allow
            if ( ( section.equals("fa") || section.equals("se") ) && bursar && (operation.equals("read") ||
operation.equals("save") )) result=true;


            System.out.println("    Allow:"+result);

            return result;
    }



    /**
     * main routine to start the service.
     * Usage: java -Djava.library.path=. SecurityCenter [-s IP[:port]/server_name] [-u
umm_specification_url]
     */
    public static void main(String[] args)
    {
        if(args.length != 0 && args.length != 2 && args.length != 4)
        {
            System.out.println("Usage: java -Djava.library.path=. SecurityCenter[-s IP[:port]/server_name] [-u
umm_specification_url]");
            return;
        }

        String url = null; //host and binding name for the component
        String ummSpecURL = null; //file url for the UMM specification of the component

        if(args.length != 0)
        {
            for(int i = 0; i < args.length; i = i + 2)
            {
                if(args[i].trim().equals("-s"))
                {
                    url = "//" + args[i+1];
                }
                else if(args[i].trim().equals("-u"))
                {
                    ummSpecURL = args[i+1];
                }
            }
        }
        if(url == null)
        {
            url = "//localhost/SecurityCenter";
        }

        if(ummSpecURL == null)
        {
            ummSpecURL = "../srs_xml/security_center_spec.xml";
        }

        try
        {
            SecurityCenter securityCenter= new SecurityCenter();
            securityCenter.setUMMSpecURL(ummSpecURL);
            Naming.rebind(url, securityCenter);
            System.out.println("Java SecurityCenteris ready:");
            System.out.println("   Host: " + url);
            System.out.println("   UMM Specification URL: " + ummSpecURL);
        }
        catch(RemoteException e)
        {
            System.out.println(e);
        }
        catch(MalformedURLException e)
        {
            System.out.println(e);
        }
    }
}
```

Test.java

```
import java.rmi.*;
import java.rmi.server.UnicastRemoteObject;
import java.net.MalformedURLException;
import java.text.NumberFormat;
import java.io.*;
import java.util.*;

/**
 * This class provides a means for composing and testing the
 * student information system case study systems
 *
 * @author Alex Crespi
 * @date October 2004
 * @version 1.0
 */
public class Test
{
    private ISecurityCenter securityCenter;
    private IStudentRecord userTerminal;
    private String securityCenter_Name;
    private String userTerminal_Name;
    private String recordServer_Name;
    private String academicRecordServer_Name;
    private String financialAidServer_Name;
    private String studentEmploymentServer_Name;

    private String securityCenter_Type;
    private String userTerminal_Type;
    private String recordServer_Type;
    private String academicRecordServer_Type;
    private String financialAidServer_Type;
    private String studentEmploymentServer_Type;

    public PrintWriter outputFile;

    /**
     * Constructor.
     */
    public Test(String ut,String rs,String ars,String fas,String ses,String sc, String output)
    {
            userTerminal_Name=ut;
            recordServer_Name=rs;
            academicRecordServer_Name=ars;
            financialAidServer_Name=fas;
            studentEmploymentServer_Name=ses;
            securityCenter_Name=sc;

            userTerminal_Type="UserTerminal";
            recordServer_Type="StudentRecordServer";
            academicRecordServer_Type="AcademicRecordServer";
            financialAidServer_Type="FinancialAidServer";
            studentEmploymentServer_Type="StudentEmploymentServer";
            securityCenter_Type="SecurityCenter";

            try{
                    outputFile=new PrintWriter(new FileWriter(new File(output)));

                    outputFile.println("System Components");
                    outputFile.println("---------------------------------------------");
                    outputFile.println("UserTerminal: "+userTerminal_Name);
                    outputFile.println("RecordServer: "+recordServer_Name);
                    outputFile.println("AcademicRecordServer: "+academicRecordServer_Name);
                    outputFile.println("FinancialAidServer: "+financialAidServer_Name);
                    outputFile.println("StudentEmploymentServer: "+studentEmploymentServer_Name);
                    outputFile.println("SecurityCenter: "+securityCenter_Name);
                    outputFile.println("\n\n");


            }catch(Exception e){
                    System.out.println("ERROR:"+e);
            }
```

```
    }

    public void printlnToFile(String s){
            try{
                    outputFile.println(s);
            }catch(Exception e){
                    System.out.println("ERROR:"+e);
            }
    }

    public void printToFile(String s){
            try{
                    outputFile.print(s);
            }catch(Exception e){
                    System.out.println("ERROR:"+e);
            }
    }

    public void configure() throws RemoteException,NotBoundException,MalformedURLException
    {

                    IConfigure configureUserTerminal=
            (IConfigure)Naming.lookup("rmi://"+userTerminal_Name);
                    IConfigure configureRecordServer=
            (IConfigure)Naming.lookup("rmi://"+recordServer_Name);
                    IConfigure configureAcademicRecordServer=
            (IConfigure)Naming.lookup("rmi://"+academicRecordServer_Name);
                    IConfigure configureFinancialAidServer=
            (IConfigure)Naming.lookup("rmi://"+financialAidServer_Name);
                    IConfigure configureStudentEmploymentServer=
            (IConfigure)Naming.lookup("rmi://"+studentEmploymentServer_Name);


                    //connect to all components for configuration

                    //configure all components to interact as a system
                    configureUserTerminal.configure(recordServer_Name,recordServer_Type);
                    configureUserTerminal.configure(securityCenter_Name,securityCenter_Type);


            configureRecordServer.configure(academicRecordServer_Name,academicRecordServer_Type);
                    configureRecordServer.configure(financialAidServer_Name,financialAidServer_Type);

            configureRecordServer.configure(studentEmploymentServer_Name,studentEmploymentServer_T
ype);
                    configureRecordServer.configure(securityCenter_Name,securityCenter_Type);

                    configureAcademicRecordServer.configure(securityCenter_Name,securityCenter_Type);
            configureStudentEmploymentServer.configure(securityCenter_Name,securityCenter_Type);
                    configureFinancialAidServer.configure(securityCenter_Name,securityCenter_Type);

    }
```

```
   public IStudentRecord getUserTerminal()throws
RemoteException,NotBoundException,MalformedURLException
   {
                  return (IStudentRecord )Naming.lookup("rmi://"+userTerminal_Name);

   }

   /**
    * main routine to start the service.
    * Usage: java Test1 [UserTerminal] [RecordServer] [AcademicRecordServer] [FinancialAidServer]
[StudentEmploymentServer] [SecurityCenter] [output file]
    */
   public static void main(String[] args)
   {
      if(args.length != 7)
      {
         System.out.println("Usage: java Test1 [UserTerminal] [RecordServer] [AcademicRecordServer]
[FinancialAidServer] [StudentEmploymentServer] [SecurityCenter] [output file]");
         return;
      }

           String ut=args[0];
           String rs=args[1];
           String ars=args[2];
           String fas=args[3];
           String ses=args[4];
           String sc=args[5];
           String output=args[6];

           //Instantiate test object for requested remote components
           Test test=new Test(ut,rs,ars,fas,ses,sc,output);


           //Configure all remote components
           try{
                  test.configure();
           }catch(Exception e) {
                  System.out.println("ERROR:"+e);
                  return;
           }

           ArrayList principals=new ArrayList();

           //read in user security attribute lists
           SecurityAttributeList sal_acrespi=new SecurityAttributeList();

           //create access id for acrespi
           ArrayList ids=new ArrayList();
           ids.add("acrespi");
           SecurityAttribute accessID=new SecurityAttribute("accessid","UniFrame",ids);
           sal_acrespi.addAttribute(accessID);

           //create roles for acrespi
           ArrayList rls=new ArrayList();
```

```
rls.add("student");
SecurityAttribute roles=new SecurityAttribute("roles","UniFrame",rls);
sal_acrespi.addAttribute(roles);

//create groups for acrespi
ArrayList gps=new ArrayList();
gps.add("students");
SecurityAttribute groups=new SecurityAttribute("groups","UniFrame",gps);
sal_acrespi.addAttribute(groups);

//add acrespi to the list of pricipals
principals.add(sal_acrespi);

//read in user security attribute lists
SecurityAttributeList sal_hblake=new SecurityAttributeList();

//create access id for hblake
ids=new ArrayList();
ids.add("hblake");
accessID=new SecurityAttribute("accessid","UniFrame",ids);
sal_hblake.addAttribute(accessID);

//create roles for hblake
rls=new ArrayList();
rls.add("student");
roles=new SecurityAttribute("roles","UniFrame",rls);
sal_hblake.addAttribute(roles);

//create groups for hblake
gps=new ArrayList();
gps.add("students");
groups=new SecurityAttribute("groups","UniFrame",gps);
sal_hblake.addAttribute(groups);

//add hblake to the list of pricipals
principals.add(sal_hblake);



//read in user security attribute lists
SecurityAttributeList sal_fburns=new SecurityAttributeList();

//create access id for fburns
ids=new ArrayList();
ids.add("fburns");
accessID=new SecurityAttribute("accessid","UniFrame",ids);
sal_fburns.addAttribute(accessID);

//create roles for fburns
rls=new ArrayList();
rls.add("bursar");
roles=new SecurityAttribute("roles","UniFrame",rls);
sal_fburns.addAttribute(roles);
```

```
//create groups for fburns
gps=new ArrayList();
gps.add("staff");
groups=new SecurityAttribute("groups","UniFrame",gps);
sal_fburns.addAttribute(groups);

//add fburns to the list of pricipals
principals.add(sal_fburns);

//read in user security attribute lists
SecurityAttributeList sal_cwinchester=new SecurityAttributeList();

//create access id for cwinchester
ids=new ArrayList();
ids.add("cwinchester");
accessID=new SecurityAttribute("accessid","UniFrame",ids);
sal_cwinchester.addAttribute(accessID);

//create roles for cwinchester
rls=new ArrayList();
rls.add("academicadvisor");
roles=new SecurityAttribute("roles","UniFrame",rls);
sal_cwinchester.addAttribute(roles);

//create groups for cwinchester
gps=new ArrayList();
gps.add("staff");
groups=new SecurityAttribute("groups","UniFrame",gps);
sal_cwinchester.addAttribute(groups);

//add cwinchesterto the list of pricipals
principals.add(sal_cwinchester);


System.out.print("acrespi Security Attributes:"+sal_acrespi+"\n");
System.out.print("hblake Security Attributes:"+sal_hblake+"\n");
System.out.print("fburns Security Attributes:"+sal_fburns+"\n");
System.out.print("cwinchesterSecurity Attributes:"+sal_cwinchester+"\n");

//print principals to output file
test.printlnToFile("Principal Security Attributes:");
test.printlnToFile("--------------------------------------------------------");
for(int i=0;i<principals.size();i++){
        SecurityAttributeList sal=(SecurityAttributeList)principals.get(i);
        test.printlnToFile(sal.toString());
}



try{
        IStudentRecord userTerminal = test.getUserTerminal();

        ArrayList students=new ArrayList();
```

```
                    //add a student for each student in the principals list
                    test.printlnToFile("Create Student Record for each Student");
                    test.printlnToFile("---------------------------------------------");
                    test.printlnToFile("PRINCIPAL;COMMAND;STUDENT;RESULT");


                    for (int i=0; i<principals.size();i++){
                            SecurityAttributeList sal=(SecurityAttributeList)principals.get(i);
                            boolean foundStudent=false;

                            for(int k=0; k<sal.size(); k++){
                                    SecurityAttribute sa=sal.getAttribute(k);
                                    if (sa.attributeType.equals("roles")){
                                            ArrayList value=(ArrayList)sa.value;
                                            for(int c=0;c<value.size();c++){

            if(((String)((ArrayList)value).get(c)).equals("student")){
                                                            foundStudent=true;
                                                    }
                                            }
                                    }
                            }
                            if(foundStudent){
                                    for(int k=0; k<sal.size(); k++){
                                            SecurityAttribute sa=sal.getAttribute(k);
                                            if (sa.attributeType.equals("accessid")){
                                                    ArrayList value=(ArrayList)sa.value;
                                                    students.add((String)((ArrayList)value).get(0));
                                                    boolean result =
userTerminal.addStudent((String)((ArrayList)value).get(0),sal);

            System.out.println((String)((ArrayList)value).get(0)+"; CREATE ;
"+(String)((ArrayList)value).get(0)+" ; "+result);

            test.printlnToFile((String)((ArrayList)value).get(0)+"; CREATE ;
"+(String)((ArrayList)value).get(0)+" ; "+result);

                                            }
                                    }
                            }
                    }


                    //submitt all possible read record combinations. each principal reads each student's record
                    test.printlnToFile("\n\nEach principal attempts to read each student record");
                    test.printlnToFile("----------------------------------------------------");
                    test.printlnToFile("PRINCIPAL;COMMAND;STUDENT;GSI;AR;FA;SE");

                    for (int i=0; i<principals.size();i++){
                            SecurityAttributeList sal = (SecurityAttributeList)principals.get(i);
                            String user="NONE";

                            for(int k=0; k<sal.size(); k++){
                                    SecurityAttribute sa=sal.getAttribute(k);
```

```
                        if (sa.attributeType.equals("accessid")){
                                ArrayList value=(ArrayList)sa.value;
                                user=(String)((ArrayList)value).get(0);
                        }
                }

                for (int k=0; k<students.size();k++){
                        String student=(String)students.get(k);
                        StudentRecord sr=userTerminal.readStudentRecord(student,sal);
                        System.out.println(user+"; READ ;"+student+";"+sr);
                        GeneralStudentInformation gsi = sr.getGeneralStudentInformation();
                        AcademicRecord ar = sr.getAcademicRecord();
                        FinancialAid fa = sr.getFinancialAid();
                        StudentEmployment se = sr.getStudentEmployment();
                        test.printToFile(user+"; READ ;"+student+" ; ");
                        if (gsi==null){test.printToFile("BLOCKED ; ");} else
{test.printToFile("ACCESSED ; ");}
                        if (ar==null){test.printToFile("BLOCKED ; ");} else
{test.printToFile("ACCESSED ; ");}
                        if (fa==null){test.printToFile("BLOCKED ; ");} else
{test.printToFile("ACCESSED ; ");}
                        if (se==null){test.printlnToFile("BLOCKED");} else
{test.printlnToFile("ACCESSED");}
                }
        }


        //submit all possible save record combinations
        test.printlnToFile("\n\nEach principal attempts to save possible configuration of each
student record");
        test.printlnToFile("--------------------------------------------------------------------------------");
        test.printlnToFile("PRINCIPAL;COMMAND;STUDENT;GSI;AR;FA;SE;RESULT");

        for (int i=0; i<principals.size();i++){
                SecurityAttributeList sal = (SecurityAttributeList)principals.get(i);
                String user="NONE";

                for(int k=0; k<sal.size(); k++){
                        SecurityAttribute sa=sal.getAttribute(k);
                        if (sa.attributeType.equals("accessid")){
                                ArrayList value=(ArrayList)sa.value;
                                user=(String)((ArrayList)value).get(0);
                        }
                }

                for (int k=0; k<students.size();k++){
                        String student=(String)students.get(k);

                        GeneralStudentInformation gsi = null;
                        for(int gcount=0;gcount<2;gcount++){
                                if(gcount==1) gsi=new GeneralStudentInformation(student);
                                AcademicRecord ar = null;
                                for(int acount=0;acount<2;acount++){
                                        if(acount==1) ar=new AcademicRecord(student);
```

```
                                                            FinancialAid fa = null;
                                                            for(int fcount=0;fcount<2;fcount++){
                                                                    if(fcount==1) fa=new
FinancialAid(student);
                                                                    StudentEmployment se = null;
                                                                    for(int ecount=0;ecount<2;ecount++){
                                                                            if(ecount==1) se=new
StudentEmployment(student);
                                                                            StudentRecord sr = new
StudentRecord(student,gsi,ar,fa,se);
                                                                            test.printToFile(user+"; SAVE
;"+student+" ; ");
                                                                            if (gsi==null){test.printToFile("NO
; ");} else {test.printToFile("YES ; ");}
                                                                            if (ar==null){test.printToFile("NO
; ");} else {test.printToFile("YES ; ");}
                                                                            if (fa==null){test.printToFile("NO
; ");} else {test.printToFile("YES ; ");}
                                                                            if (se==null){test.printToFile("NO
; ");} else {test.printToFile("YES ; ");}
System.out.println(sal);
System.out.println(sr);
                                                                            boolean result =
userTerminal.saveStudentRecord(student,sr ,sal);
System.out.println(result);
System.out.println("------");
                                                                            if (result)
{test.printlnToFile("SUCCEEDED");} else {test.printlnToFile("FAILED");}
                                                                    }
                                                            }
                                                    }
                                            }
                                    }
                            }



                }catch(Exception e) {
                        System.out.println("ERROR:"+e);
                        test.outputFile.close();

                        return;
                }

                test.outputFile.close();
    }
}
```

Remote Interfaces

```
import java.rmi.*;

public interface IAcademicRecord extends Remote, ITransaction {
	boolean addStudent(String sid, int tid, SecurityAttributeList sal) throws RemoteException;
	AcademicRecord readAcademicRecord(String sid, SecurityAttributeList sal) throws
RemoteException;
	boolean saveAcademicRecord(String sid,AcademicRecord se, int tid, SecurityAttributeList sal)
throws RemoteException;
}



import java.rmi.*;
import java.util.*;

public interface IConfigure extends Remote
{
	void configure(String value, String option) throws RemoteException;
	void deconfigure(String value, String option) throws RemoteException;

	Object list_configuration(String option) throws RemoteException;
	void reset_configuration() throws RemoteException;

	String getUmmSpecURL() throws RemoteException;
	void setUMMSpecURL(String url) throws RemoteException;

	String getLogicSpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException;

	String getTLASpecification(String componentName, Hashtable postProcessingCollaborators)
throws RemoteException;
}



import java.rmi.*;

public interface IFinancialAid extends Remote, ITransaction {
	boolean addStudent(String sid, int tid, SecurityAttributeList sal) throws RemoteException;
	FinancialAid readFinancialAid(String sid, SecurityAttributeList sal) throws RemoteException;
	boolean saveFinancialAid(String sid,FinancialAid fa, int tid, SecurityAttributeList sal) throws
RemoteException;
}



import java.rmi.*;

public interface ISecurityCenter extends Remote {
	boolean evaluatePolicy(SecurityAttributeList sal, ResourceName resource, String operation)
throws RemoteException;
}


import java.rmi.*;

public interface IStudentEmployment extends Remote, ITransaction {
	boolean addStudent(String sid, int tid, SecurityAttributeList sal) throws RemoteException;
	StudentEmployment readStudentEmployment(String sid, SecurityAttributeList sal) throws
RemoteException;
	boolean saveStudentEmployment(String sid,StudentEmployment se, int tid, SecurityAttributeList
sal) throws RemoteException;
}


import java.rmi.*;
import java.util.*;

public interface IStudentRecord extends Remote{

	boolean addStudent(String sid, SecurityAttributeList sal) throws RemoteException;
	StudentRecord readStudentRecord(String sid, SecurityAttributeList sal) throws RemoteException;
	boolean saveStudentRecord(String sid,StudentRecord sr, SecurityAttributeList sal) throws
RemoteException;
}


import java.rmi.*;

public interface ITransaction extends Remote {
	int transaction(SecurityAttributeList sal) throws RemoteException;
	void commit(int tid, SecurityAttributeList sal) throws RemoteException;
	void rollback(int tid, SecurityAttributeList sal) throws RemoteException;

}
```

Appendix G: Component Search Simulation Implementation

This appendix includes the code used to implement testing of the component matching algorithm including the access control properties. It includes code for the headhunter, code to generate component Prolog access control specifications, and code for running and testing the headhunter.

Headhunter Code

```
import java.net.*;
import java.util.*;
import java.sql.*;

import java.rmi.*;
import java.rmi.server.*;
import java.rmi.registry.*;
import java.net.URL;

import java.security.*;
import java.io.*;
import java.lang.*;

import bprolog.plc.Plc;

/**
 * Insert the type's description here.
 * Creation date: (10/15/2001 11:50:10 AM)
 * @author: Nanditha Nayani
 */

public class Headhunter
{

private Hashtable registryTable = new Hashtable();
private java.lang.String userType = "Headhunter";

//newly added hashtable
private Hashtable resultTable= new Hashtable();

//Component Retrieval Time measurement
private long start_time,end_time;
private long elapse_time;

//newly added ArrayList to hold query ID's
private ArrayList QIDList=new ArrayList();

//newly added strings to pass data to QueryBean
```

```
private String componentTableName=null;

//private String functionTableName=null;
//newly added for Grid
private static String forGrid = null;
private String ummspecurl = null;

private Plc prologEngine = new Plc();

public Hashtable performSearch(QueryBean querybean)
{
        long time1,time2,time3,time4,time5,time6,time7,time8;

        System.out.println("Starting to process query request");

        time1 = System.currentTimeMillis();

        MetaRepositoryHelper srchEngine = new MetaRepositoryHelper(querybean);

        time2 = System.currentTimeMillis();
        try
        {
        try {

                        resultTable=srchEngine.getSearchResultTable(componentTableName);
                }
        catch(Exception e) {
                System.out.println("  No record found matching search criteria in local repository");
                resultTable.clear();
                }

                //System.out.println("Not out of loop");
                if(resultTable.isEmpty()==true)
                {
                        System.out.println("  Components not found int the metarepository of
primaryheadhunter");
                }
                else
        {
          System.out.println("  Components found in the metarepository of headhunter");
        }
        }  //end try

        catch (Exception e)
        {
                System.out.println(e);
                return null;
        }


        time3 = System.currentTimeMillis();

        System.out.println("  Filtering Components Through Access Control Requirements.");
```

```
            System.out.println("\n   COMPONENTS BEFORE ACCESS CONTROL FILTERING:");
            Enumeration keys = resultTable.keys();
            while (keys.hasMoreElements())
           {
                        Object key = keys.nextElement();
                        ConcreteComponent component = (ConcreteComponent) resultTable.get(key);
                        System.out.println("    "+component.getID());
               }//end while

            time4 = System.currentTimeMillis();

            //APPEND ON FACTS FOR REQUIRED PROTECTED RESOURCES
            StringBuffer prologDatabase=new StringBuffer();
            StringBuffer prologConsider=new StringBuffer();
            StringBuffer prologContain=new StringBuffer();
            StringBuffer prologEnsureProtect=new StringBuffer();

            prologDatabase.append("require_protect([");
            String[] RequiredResources = querybean.getRequiredProtectedResources();

            for(int i=0; i<RequiredResources.length;i++){
                        if (i!=0) prologDatabase.append(",");
                        prologDatabase.append("'"+RequiredResources[i]+"'");
            }
            prologDatabase.append("]).\n");


            //LOOP THROUGH ALL ELEMENTS IN HASHTABLE TO BUILD THE PROLOG
            keys = resultTable.keys();
            while (keys.hasMoreElements())
           {
                        ConcreteComponent component = (ConcreteComponent)
resultTable.get(keys.nextElement());

                        //ADD A FACT TO CONSIDER THIS COMPONENT
                        prologConsider.append("consider('"+component.getID()+"').\n");


                        //ADD FACTS ABOUT WHAT RESOURCES THIS COMPONENT CONTAINS
                        String[] contains = component.getContainedResources();
                        if (contains.length>0){
                                    for(int i=0; i<contains.length;i++){

            prologContain.append("contain('"+component.getID()+"','"+contains[i]+"').\n");

                                    }
                        }

                        //ADD FACTS ABOUT WHAT RESOURCES THIS COMPONENT PROTECTS
                        String[] protects = component.getProtectedResources();
                        if (protects.length>0){
                                    for(int i=0; i<protects.length;i++){
            prologContain.append("ensure_protect('"+component.getID()+"','"+protects[i]+"').\n");
```

```
                    }
                } else {
                    prologContain.append("ensure_protect('"+component.getID()+"','').\n");

                }

            }//end while

        //APPEND ALL consider('componentID'). FACTS TO DATABASE
        prologDatabase.append(prologConsider);
        prologDatabase.append("\n");

        //APPEND ALL contain('componentID','resourceName'). FACTS TO DATABASE
        prologDatabase.append(prologContain);
        prologDatabase.append("\n");

        //APPEND ALL ensure_protect('componentID','resourceName'). FACTS TO DATABASE
        prologDatabase.append(prologEnsureProtect);
        prologDatabase.append("\n");

        //APPEND MATCHING RULES TO THE DATABASE
        prologDatabase.append("protects(_,[]).\n");
        prologDatabase.append("protects(Component,[Resource|OtherResources]) :-\n");
        prologDatabase.append("    (contain(Component,Resource),ensure_protect(Component,Resource)
; not contain(Component,Resource)),\n");
        prologDatabase.append("     protects(Component,OtherResources).\n\n");

        prologDatabase.append("match(Component) :- consider(Component),\n");
        prologDatabase.append("                 require_protect(Resources),\n");
        prologDatabase.append("                 protects(Component,Resources).\n\n");

        time5 = System.currentTimeMillis();

        //SAVE PROLOG DATABASE TO FILE FOR PROCESSING
        try{
            File f=new File("prologMatchingDB.pl");
            FileWriter fw=new FileWriter(f);
            PrintWriter pw=new PrintWriter(fw,true);
            pw.println(prologDatabase.toString());
            pw.flush();
            pw.close();

        }catch(IOException e){System.out.println("ERROR:"+e);}

        time6 = System.currentTimeMillis();
System.out.println(prologDatabase.toString());
        //RUN PROLOG ENGINE
//System.out.println("before start");
//      Plc.startPlc(new String[] {});

        prologEngine.exec("consult('prologMatchingDB')");


        time7 = System.currentTimeMillis();
```

```
//LOOP THROUGH QUERYING FOR EACH COMPONENT IN THE HASHTABLE
//REMOVING COMPONENTS THAT DO NOT MATCH ACCESS CONTROL REQUIREMENTS.
            keys = resultTable.keys();
            while (keys.hasMoreElements())
          {
                        Object key = keys.nextElement();
                        ConcreteComponent component = (ConcreteComponent) resultTable.get(key);
                        String query="match('"+component.getID()+"')";


                        if(!prologEngine.exec(query))
                                    resultTable.remove(key);
               }//end while


            time8 = System.currentTimeMillis();

            //DISPLAY ALL COMPONENTS SURVIVING ACCESS CONTROL FILTER
            System.out.println("\n   COMPONENTS AFTER ACCESS CONTROL FILTERING:");
            keys = resultTable.keys();
            while (keys.hasMoreElements())
          {
                        Object key = keys.nextElement();
                        ConcreteComponent component = (ConcreteComponent) resultTable.get(key);
                        System.out.println("    "+component.getID());
               }//end while

            System.out.println("\n  Access Control Filtering Done.");
            return resultTable;

}//end function


//newly written method to search for components
public Hashtable performComponentSearch(QueryBean querybean)
{
            System.out.println("\nProcessing Query request...");
            MetaRepositoryHelper srchEngine=new MetaRepositoryHelper(querybean);
            try
            {
                        return srchEngine.getSearchResultTable(componentTableName);
            }
            catch(Exception e)
            {
                        //System.out.println("Error in searching Local Metarepository"+ e.getMessage());
          //e.printStackTrace();
                        return null;
            }
}

private void createMetaRepository(String userName) throws Exception {

     SQLHelper sqlEngine = new SQLHelper();
     String dropUmmSpecs = "DROP TABLE "+userName+"UMMSpecification";
String dropAlgorithms = "DROP TABLE "+userName+"Algorithms";
String dropreqdinterface = "DROP TABLE "+userName+"RequiredInterfaces";
String dropProvidedInterfaces = "DROP TABLE "+userName+"ProvidedInterfaces";
String dropTechnologies = "DROP TABLE "+userName+"Technologies";
String dropExpectedResources = "DROP TABLE "+userName+"ExpectedResources";
String dropDesignPatterns = "DROP TABLE "+userName+"DesignPatterns";
String dropKnownUsages = "DROP TABLE "+userName+"KnownUsages";
String dropAliases = "DROP TABLE "+userName+"Aliases";
String dropPreProcessing = "DROP TABLE "+userName+"PreProcessing";
String dropPostProcessing = "DROP TABLE "+userName+"PostProcessing";
String dropContainedResources = "DROP TABLE "+userName+"ContainedResources";
String dropProtectedResources = "DROP TABLE "+userName+"ProtectedResources";
String dropCompFuncQoS = "DROP TABLE "+userName+"CompFuncQoS";

System.out.println("\nMetaRepository being created....");

try {
	sqlEngine.updateTable(dropUmmSpecs);
}catch (Exception e){}
try {
	sqlEngine.updateTable(dropAlgorithms);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropreqdinterface);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropProvidedInterfaces);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropTechnologies);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropExpectedResources);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropDesignPatterns);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropKnownUsages);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropAliases);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropPreProcessing);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropPostProcessing);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropContainedResources);
}catch (Exception e){}
try {
		sqlEngine.updateTable(dropProtectedResources);
```

```
}catch (Exception e){}
try {
                sqlEngine.updateTable(dropCompFuncQoS);
}catch (Exception e)
{
  //System.out.println("Exception in dropping table"+e.getMessage());
}

      String createUMMSpecification = "create table "+ userName + "UMMSpecification" +
                "(ComponentName VARCHAR(256), " +
                "SubCase VARCHAR(256), " +
                "DomainName VARCHAR(256), " +
                "SystemName  VARCHAR(256), " +
                "Description VARCHAR(2000), " +
                "Id VARCHAR(256) PRIMARY KEY, " +
                "Version VARCHAR(256), " +
                "Author VARCHAR(256), " +
                "CreatingDate VARCHAR(256), " +
                "Validity VARCHAR(256), " +
                "Atomicity VARCHAR(256), " +
                "Registration VARCHAR(256), " +
                "Model VARCHAR(256), " +
                "Purpose VARCHAR(2000), " +
                "Complexity VARCHAR(256), " +
                "Mobility VARCHAR(256), " +
                "Security VARCHAR(256), " +
                "FaultTolerance VARCHAR(256), " +
                "QosLevel VARCHAR(256), " +
                "Cost VARCHAR(256), " +
                "QualityLevel VARCHAR(256))";

    String createAlgorithms = "Create table "+ userName + "Algorithms" +
                "(Id VARCHAR(256) NOT NULL, " +
                            "ComponentName VARCHAR(256), " +
                "DomainName VARCHAR(256), " +
                "SystemName VARCHAR(256), " +
                "Algorithm VARCHAR(256) NOT NULL," +
                " PRIMARY KEY (Id, Algorithm))";

    String createRequiredInterfaces = "Create table "+ userName + "RequiredInterfaces" +
                "(Id VARCHAR(256) NOT NULL, " +
                "ComponentName VARCHAR(256), " +
                "DomainName VARCHAR(256), " +
                "SystemName VARCHAR(256), " +
                "Interface VARCHAR(256) NOT NULL, " +
                                        " PRIMARY KEY (Id, Interface))";


    String createProvidedInterfaces = "create table "+ userName + "ProvidedInterfaces" +
                "(Id VARCHAR(256) NOT NULL, " +
                "ComponentName VARCHAR(256), " +
                "DomainName VARCHAR(256), " +
                "SystemName VARCHAR(256), " +
                "Interface VARCHAR(256) NOT NULL, " +
            " PRIMARY KEY (Id, Interface))";

String createTechnologies = "create table "+ userName + "Technologies" +
            "(Id VARCHAR(256) NOT NULL, " +
            "ComponentName VARCHAR(256), " +
            "DomainName VARCHAR(256), " +
            "SystemName VARCHAR(256), " +
            "Technology VARCHAR(256) NOT NULL," +
            " PRIMARY KEY (Id, Technology))";

String createExpectedResources = "create table "+ userName + "ExpectedResources" +
                        "(Id VARCHAR(256) NOT NULL, " +
            "ComponentName VARCHAR(256), " +
            "DomainName VARCHAR(256), " +
            "SystemName VARCHAR(256), " +
            "ExpectedResource VARCHAR(256) NOT NULL," +
            " PRIMARY KEY (Id, ExpectedResource))";

String createDesignPatterns = "create table "+ userName + "DesignPatterns" +
                        "(Id VARCHAR(256) NOT NULL, " +
            "ComponentName VARCHAR(256), " +
            "DomainName VARCHAR(256), " +
            "SystemName VARCHAR(256), " +
            "Pattern VARCHAR(256) NOT NULL," +
            " PRIMARY KEY (Id, Pattern))";

String createKnownUsages = "create table "+ userName + "KnownUsages" +
                        "(Id VARCHAR(256) NOT NULL, " +
            "ComponentName VARCHAR(256), " +
            "DomainName VARCHAR(256), " +
            "SystemName VARCHAR(256), " +
            "Usage VARCHAR(256) NOT NULL," +
            " PRIMARY KEY (Id, Usage))";

String createAliases = "create table "+ userName + "Aliases" +
            "(Id VARCHAR(256) NOT NULL, " +
            "ComponentName VARCHAR(256), " +
            "DomainName VARCHAR(256), " +
            "SystemName VARCHAR(256), " +
            "Alias VARCHAR(256) NOT NULL," +
            " PRIMARY KEY (Id, Alias))";

String createPreProcessingCollaborators = "create table "+ userName + "PreProcessing" +
            "(Id VARCHAR(256) NOT NULL, " +
            "ComponentName VARCHAR(256), " +
            "DomainName VARCHAR(256), " +
            "SystemName VARCHAR(256), " +
            "Collaborator VARCHAR(256) NOT NULL," +
            " PRIMARY KEY (Id, Collaborator))";

String createPostProcessingCollaborators = "create table "+ userName + "PostProcessing" +
            "(Id VARCHAR(256) NOT NULL, " +
            "ComponentName VARCHAR(256), " +
            "DomainName VARCHAR(256), " +
                "SystemName VARCHAR(256), " +
                "Collaborator VARCHAR(256) NOT NULL," +
                " PRIMARY KEY (Id, Collaborator))";

        //TABLES FOR ACCESS CONTROL RELATED DATA
        String createContainedResources = "create table "+ userName + "ContainedResources" +
                "(Id VARCHAR(256) NOT NULL, " +
                "ComponentName VARCHAR(256), " +
                "DomainName VARCHAR(256), " +
                "SystemName VARCHAR(256), " +
                "ResourceName VARCHAR(256) NOT NULL," +
                " PRIMARY KEY (Id, ResourceName))";

        String createProtectedResources = "create table "+ userName + "ProtectedResources" +
                "(Id VARCHAR(256) NOT NULL, " +
                "ComponentName VARCHAR(256), " +
                "DomainName VARCHAR(256), " +
                "SystemName VARCHAR(256), " +
                "ResourceName VARCHAR(256) NOT NULL," +
                " PRIMARY KEY (Id, ResourceName))";


// In the following table, the length of the columns of the tables have been changed, since Oracle doesn't
allow
// the length of the primary key to exceed 756 key length

        String createqoscompfunc = "create table "+ userName + "CompFuncQoS" +
                "(Id VARCHAR(100) NOT NULL, " +
                "ComponentName VARCHAR(256), " +
                "SystemName VARCHAR(256), " +
                "FunctionName VARCHAR(100) NOT NULL, " +
                "QoSParameter VARCHAR(100) NOT NULL, " +
                "Value VARCHAR(100) NOT NULL, " +
                " PRIMARY KEY (Id, FunctionName, QoSParameter, Value))";

        try{
            sqlEngine.updateTable(createUMMSpecification);
        } catch(Exception e) {System.out.println("UMM Spec Table problem"+e.getMessage());}
        try{
        sqlEngine.updateTable(createAlgorithms);
        }catch(Exception e){System.out.println("Algo table problem");}
      try{
         sqlEngine.updateTable(createRequiredInterfaces);
         }catch(Exception e){System.out.println("Required Interface table problem");}
       try{
         sqlEngine.updateTable(createProvidedInterfaces);
        }catch(Exception e){System.out.println("Provided Interface table problem");}
      try{
        sqlEngine.updateTable(createTechnologies);
        }catch(Exception e){System.out.println("Technology table problem");}
      try{
        sqlEngine.updateTable(createExpectedResources);
        }catch(Exception e){System.out.println("Resources table problem");}
      try{
        sqlEngine.updateTable(createDesignPatterns);
        }catch(Exception e){System.out.println("DesignPatterns table problem");}
      try{
        sqlEngine.updateTable(createKnownUsages);
        }catch(Exception e){System.out.println("knownusages table problem");}
      try{
        sqlEngine.updateTable(createAliases);
        }catch(Exception e){System.out.println("Aliases table problem"+e.getMessage());}

         //CREATE ACCESS CONTROL RESOURCES TABLES
      try{
        sqlEngine.updateTable(createContainedResources);
        }catch(Exception e){System.out.println("Contained Resources table problem"+e.getMessage());}
      try{
        sqlEngine.updateTable(createProtectedResources);
        }catch(Exception e){System.out.println("Protected Resources table problem"+e.getMessage());}
      try{
        sqlEngine.updateTable(createPreProcessingCollaborators);
        }catch(Exception e){System.out.println("Preprocessing table problem"+e.getMessage());}
      try{
        sqlEngine.updateTable(createPostProcessingCollaborators);
        }catch(Exception e){System.out.println("Postprocessing table problem"+e.getMessage());}
      try{
        sqlEngine.updateTable(createqoscompfunc);
        }catch(Exception e){System.out.println(" Problem in qos update table: "+e.getMessage());}

         sqlEngine.shutDown();

 System.out.println("MetaRepository created.");
}



public static void main(String[] args) {

         long mcastTime = 50000;
         String headhunterLocation ="n/a";
         String dsmLocation="n/a";
         int mcastPort = 10000;
         String userName = "AC";
         String password = "simulation";
         String domain = "n/a";

         Headhunter h=new Headhunter(
                             mcastTime,
                             mcastPort,
                             userName,
                             password,
                             domain,
                             headhunterLocation,
                             dsmLocation);

}
```

```
 public Headhunter(
   long mcastTime,
            int mcastPort,
            String userName,
            String password,
            String domain,
            String headhunterLocation,
            String dsmLocation)
{

            System.out.println("Headhunter activating.");
            componentTableName=userName;
            //functionTableName=userName+"Function";
            prologEngine.startPlc(new String[] {});

      try {
                        createMetaRepository(userName);
                        populateMetaRepository();
            } catch (Exception e) {
                        //Ignore if repository already created
                        System.out.println(e.getMessage());
                        System.out.println("Exception occured due to any of the following reasons:");
                        System.out.println(" Tables already exist in the database");
                        System.out.println(" Primary key constraint name and foreign key constraint name
already in use");
                        System.out.println(" Primary key constraint name and foreign key constraint name are
same");
            }

                        System.out.println("Headhunter ready.");
} //end of constructor


private void populateMetaRepository() {

     System.out.println("\nMetaRepository being populated....");

            // retrieve component info

            try {
                        Hashtable CompData = getComponentData();


                        System.out.println("  Saving Components in MR.");
                        Enumeration e = CompData.elements();
                        while (e.hasMoreElements())
           {
           ConcreteComponent component = new ConcreteComponent();
           component=(ConcreteComponent) e.nextElement();

                         try {
               buildpersist buildcomponent = new buildpersist();
                              buildcomponent.persist(component, componentTableName);
                            } catch (Exception ex)
```

```
                {
                  System.out.println("Error in persisting components:"+ex.getMessage());
                                //If the component already exists in MR then ignore.
                      }
                       System.out.println("   -saved: " + component.getID() + " into MR.");
                      }//end while

            } catch (Exception ex) {
                      System.out.println("HH exception " + ex);
            }

            System.out.println("  Done Saving Components.");
            System.out.println("....MetaRepository populated.");

}


public Hashtable getComponentData()
{
            Hashtable objectTable = new Hashtable();
            try
              {
                           System.out.println("  Loading UMM Specs");


                           //Specify which UMM XML files will be loaded

                           String [] ummFiles= {      "UserTerminal1.xml",
                                                              "RecordServer1.xml",
                                                              "AcademicRecordServer1.xml",
                                                              "FinancialAidServer1.xml",
                                                              "StudentEmploymentServer1.xml",
                                                              "UserTerminal1.xml",
                                                              "RecordServer2.xml",
                                                              "AcademicRecordServer2.xml",
                                                              "FinancialAidServer2.xml",
                                                              "StudentEmploymentServer2.xml",
                                                              "RecordServer3.xml",
                                                              "RecordServer4.xml",
                                                };

                           for(int i=0;i<ummFiles.length;i++)
                        {
            //Call the XMLParser by passing this URL. The XML Parser will parse the XML specification
            //and construct a Component from the specification which it returns.

                                 UniFrameSpecificationParser xmlDomParser = new
UniFrameSpecificationParser(ummFiles[i]);
                                 ConcreteComponent component =
xmlDomParser.getConcreteComponent();

                                 //Add the component to the Hashtable
                                  objectTable.put(ummFiles[i],component);
                        }//end for

                }
                catch(Exception e){
                        System.out.println(e.getMessage());
                }

        System.out.println("  Done Loading UMM Specs.");

        //Once the Hashtable is filled return the hashtable
        return objectTable;
}

}//end of HeadHunter
```

PrologACModel.java

Acts as a simulation of components during searching by providing prolog access control model code for each component.

```
import java.util.*;
import java.io.*;


/**
 * Insert the type's description here.
 * Creation date: (10/15/2001 11:50:10 AM)
 * @author: Nanditha Nayani
 */

public class PrologACModel
{

        public PrologACModel(){


        }


public String getSystemModel(){
        StringBuffer prologBuffer = new StringBuffer();
        Date d=new Date();
        prologBuffer.append("% FILE: studentRecordPrologACModel.pl\n");
        prologBuffer.append("% DATE: "+d.toString()+"\n");
        prologBuffer.append("% CREATED BY: UniFrame System Generator\n");
        prologBuffer.append("% SYSTEM: Student Record System Access Control Model\n\n");
        prologBuffer.append("%**********FACTS and PREDICATES for student records
system**********\n");
```

```
        prologBuffer.append("%*** These facts define system users and their security credentials
***\n");
        prologBuffer.append("authenticate(acrespi,credential(accessID(acrespi),role([student]),group([stu
dent]))).\n");
        prologBuffer.append("authenticate(hblake,credential(accessID(hblake),role([student]),group([stud
ent]))).\n");
        prologBuffer.append("authenticate(fburns,credential(accessID(fburns),role([bursar]),group([staff])
)).\n");
        prologBuffer.append("authenticate(cwinchester,credential(accessID(cwinchester),role([academica
dvisor]),group([staff]))).\n\n");

        prologBuffer.append("%*** This predicate authenticates a principal in the system ***\n");
        prologBuffer.append("principal(AID,P):- authenticate(AID,P).\n\n");

        prologBuffer.append("%*** This predicate identifies students in the system ***\n");
        prologBuffer.append("student(S):-
authenticate(S,P),arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),member(student,ROLE).\n\n");

        prologBuffer.append("%*** This predicate is used to represent allowed access to a resource
***\n");
        prologBuffer.append("true(R,O):-write('ALLOWED:'),write(R),write(O),nl.\n\n");

        prologBuffer.append("%*** This predicate is used to represent access control decisions ***\n");
        prologBuffer.append("accessControl(P,R,O):-arg(1,P,AIDPRED),arg(1,AIDPRED,AID),\n");
        prologBuffer.append("                    arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLES),\n");
        prologBuffer.append("                    arg(3,P,GROUPPRED),arg(1,GROUPPRED,GROUP),\n");
        prologBuffer.append("                    nl,write('Security Attributes:'),\n");
        prologBuffer.append("                    nl,write('  ACCESSID='),write(AID),\n");
        prologBuffer.append("                    nl,write('  GROUP='),write(GROUP),\n");
        prologBuffer.append("                    nl,write('  ROLES='),write(ROLES),\n");
        prologBuffer.append("                    nl,write('  ResourceName='),write(R),\n");
        prologBuffer.append("\n");
        prologBuffer.append("                    %*** separate out the resource name components ***\n");
        prologBuffer.append("                    R=[SID,SECTION,PART|_],\n");
        prologBuffer.append("                    nl,nl,write('Resource:'),\n");
        prologBuffer.append("                    nl,write('  SR='),write(SID),\n");
        prologBuffer.append("                    nl,write('  Section='),write(SECTION),\n");
        prologBuffer.append("                    nl,write('  Part='),write(PART),nl,\n");
        prologBuffer.append("\n");
        prologBuffer.append("                    %*** Differentiate between students and staff ***\n");
        prologBuffer.append("                    (member(student,GROUP)->
checkstudent(ROLES,R,O);checkstaff(ROLES,R,O)).\n");
        prologBuffer.append("\n");
        prologBuffer.append("%*** evalute policy for students trying to access the records ***\n");
        prologBuffer.append("\n");
        prologBuffer.append("checkstudent(ROLES,R,O):- nl,write('checking student credentials for
access. '),nl,\n");
        prologBuffer.append("                    R=[SID,SECTION,PART|_],\n");
        prologBuffer.append("                    write('  Resource='),write(R),nl,\n");
        prologBuffer.append("                    write('  ROLES='),write(ROLES),nl,\n");
        prologBuffer.append("                    write('  Operation='),write(O),nl,\n");
        prologBuffer.append("                    ( ( (member(studentowner,ROLES),(not
SECTION==[section,notes]),O==read);\n");
```

```
		prologBuffer.append("                  
(member(studentowner,ROLES),SECTION==[section,gsi],O==save)       ) ->true(R,O)).\n");
		prologBuffer.append("\n");
		prologBuffer.append("\n");
		prologBuffer.append("%*** evaluate policy for staff try to access the records ***\n");
		prologBuffer.append("\n");
		prologBuffer.append("checkstaff(ROLES,R,O):-nl,write('check staff credentials for 
access.'),nl,\n");
		prologBuffer.append("                           R=[SID,SECTION,PART|_],\n");
		prologBuffer.append("                           write('  Resource='),write(R),nl,\n");
		prologBuffer.append("                           write('  ROLES='),write(ROLES),nl,\n");
		prologBuffer.append("                           write('  Operation='),write(O),nl,\n");
		prologBuffer.append("                           ( ( 
(member(bursar,ROLES),SECTION==[section,gsi],O==read);\n");
		prologBuffer.append("                  
(member(bursar,ROLES),SECTION==[section,se],O==read); \n");
		prologBuffer.append("                  
(member(bursar,ROLES),SECTION==[section,se],O==save);\n");
		prologBuffer.append("                  
(member(bursar,ROLES),SECTION==[section,fa],O==read);\n");
		prologBuffer.append("                  
(member(bursar,ROLES),SECTION==[section,fa],O==save);\n");
		prologBuffer.append("                  
(member(bursar,ROLES),SECTION==[section,notes],O==read); \n");
		prologBuffer.append("                  
(member(academicadvisor,ROLES),SECTION==[section,gsi],O==read);\n");
		prologBuffer.append("                  
(member(academicadvisor,ROLES),SECTION==[section,ar],O==read);\n");
		prologBuffer.append("                  
(member(academicadvisor,ROLES),SECTION==[section,ar],O==save)\n");
		prologBuffer.append("                             ) ->true(R,O)).\n");
		prologBuffer.append("\n");

		return prologBuffer.toString();
	}


	public String getComponentModel(String componentID, String componentName, Hashtable 
postProcessingCollaborators){

		StringBuffer prologBuffer = new StringBuffer();

		prologBuffer.append("%**********"+componentID+"***********\n");
		if (componentID.equals("UserTerminal1"))
		{
			//user_terminal_1_readStudentRecord(AID,S)    :-
principal(AID,P),student(S),record_server_1_readStudentRecord(P,S).
			//user_terminal_1_saveStudentRecord(AID,S,SR):-
principal(AID,P),student(S),record_server_1_saveStudentRecord(P,S,SR).

			prologBuffer.append(componentName+"_readStudentRecord(AID,S)    :-
principal(AID,P),student(S),");
			//read in record server name from post processing collaborators list
			String recordServerName = (String) postProcessingCollaborators.get("recordserver");
```

```
                    prologBuffer.append(recordServerName);
                    prologBuffer.append("_readStudentRecord(P,S).\n");

                    prologBuffer.append(componentName+"_saveStudentRecord(AID,S,SR)    :-
principal(AID,P),student(S),");

                    prologBuffer.append(recordServerName);
                    prologBuffer.append("_saveStudentRecord(P,S,SR).\n");
            }

            else if (componentID.equals("RecordServer1"))
            {

                    prologBuffer.append("%*** These facts represent student data in the record server
***\n");

                    prologBuffer.append(componentName+"_sr(acrespi).\n");
                    prologBuffer.append(componentName+"_sr(hblake).\n");
                    prologBuffer.append(componentName+"_gsi(acrespi,alex,crespi,indianapolis).\n");
                    prologBuffer.append(componentName+"_gsi(hblake,henry,blake,muncie).\n");
                    prologBuffer.append(componentName+"_note(acrespi,fburns, ['this is a note']).\n");
                    prologBuffer.append(componentName+"_note(acrespi,cwinchester,['this is another
note']).\n\n");

                    prologBuffer.append("%*** Predicate for reading student record ***\n");

//record_server_1_readStudentRecord(P,S):-
//          financial_aid_server_1_readStudentRecord(P,S) ;
//          academic_record_server_1_readStudentRecord(P,S) ;
//          student_employment_server_1_readStudentRecord(P,S) ;
//          (record_server_1_gsi(S,_,_,_),record_server_1_Guard(P,[[sr,S],[section,gsi],[all,all]],read)).

                    prologBuffer.append(componentName+"_readStudentRecord(P,S):-\n");

                    //read in record server name from post processing collaborators list
                    String financialAidServerName = (String)
postProcessingCollaborators.get("financialaidserver");
                    String academicRecordServerName = (String)
postProcessingCollaborators.get("academicrecordserver");
                    String studentEmploymentServerName = (String)
postProcessingCollaborators.get("studentemploymentserver");

                    prologBuffer.append("    "+financialAidServerName     +"_readStudentRecord(P,S) ;\n");
                    prologBuffer.append("    "+academicRecordServerName   +"_readStudentRecord(P,S)
;\n");
                    prologBuffer.append("    "+studentEmploymentServerName+"_readStudentRecord(P,S)
;\n");
                    prologBuffer.append("    "+"("+componentName+"_gsi(S,_,_,_)");
                    prologBuffer.append(").\n\n");


                    prologBuffer.append("%*** Predicate for writing student record ***\n");
```

```
//record_server_1_saveStudentRecord(P,S,SR):-
//          ((member(ar,SR) , financial_aid_server_1_saveStudentRecord(P,S,SR)); not member(ar,SR) ),
//          ((member(fa,SR) , academic_record_server_1_saveStudentRecord(P,S,SR)); not member(fa,SR)
),
//          ((member(se,SR) , student_employment_server_1_saveStudentRecord(P,S,SR)); not
member(se,SR) ).

                        prologBuffer.append(componentName+"_saveStudentRecord(P,S,SR):-\n");
                        prologBuffer.append("    "+"((member(fa,SR) ,"+financialAidServerName
+"_saveStudentRecord(P,S,SR)); not member(fa,SR) ),\n");
                        prologBuffer.append("    "+"((member(ar,SR) ,"+academicRecordServerName
+"_saveStudentRecord(P,S,SR)); not member(ar,SR) ),\n");
                        prologBuffer.append("    "+"((member(se,SR)
,"+studentEmploymentServerName+"_saveStudentRecord(P,S,SR)); not member(se,SR) ).\n\n");
            }


            else if (componentID.equals("AcademicRecordServer1"))
            {
                        prologBuffer.append("%*** These facts represent student data in the academic record
server ***\n");

            prologBuffer.append(componentName+"_ar(acrespi,dps([computer_science,business]),ts([[cs403,
a],[cs595,b]])).\n");

            prologBuffer.append(componentName+"_ar(hblake,dps([chemistry]),ts([[c101,a],[c132,a]])).\n\n"
);


                        prologBuffer.append("%*** These predicates represent the academic record server
***\n");

//academic_record_server_1_readStudentRecord(P,S):-
//                academic_record_server_1_ar(S,_,_).

//academic_record_server_1_saveStudentRecord(P,S,AR).

                        prologBuffer.append(componentName+"_readStudentRecord(P,S):- ");
                        prologBuffer.append(componentName+"_ar(S,_,_).\n");
                        prologBuffer.append(componentName+"_saveStudentRecord(P,S,AR).\n\n ");
            }

            else if (componentID.equals("FinancialAidServer1"))
            {
                        prologBuffer.append("%*** These facts represent student data in the financial aid server
***\n");

            prologBuffer.append(componentName+"_fa(acrespi,[bankofamerica,sallemay],[pell],[volleyball],[
]).\n");

            prologBuffer.append(componentName+"_fa(hblake,[bankofamerica,sallemay],[pell],[volleyball],[
]).\n\n");
//financial_aid_server_1_readStudentRecord(P,S):-
//                          financial_aid_server_1_fa(S,_,_,_,_).
```

```
//financial_aid_server_1_saveStudentRecord(P,S,SLR).

                prologBuffer.append("%*** These predicates represent the financial aid server ***\n");


                prologBuffer.append(componentName+"_readStudentRecord(P,S):- ");
                prologBuffer.append(componentName+"_fa(S,_,_,_,_).\n");

                prologBuffer.append(componentName+"_saveStudentRecord(P,S,AR).\n\n ");
        }

        else if (componentID.equals("StudentEmploymentServer1"))
        {
                prologBuffer.append("%*** These facts represent student data in the student
employment server ***\n");

        prologBuffer.append(componentName+"_se(acrespi,[[fall2003],[spring2004],[fall2004]],[]).\n");

        prologBuffer.append(componentName+"_se(hblake,[[fall2003],[spring2004],[fall2004]],[]).\n\n");

                prologBuffer.append("%*** These predicates represent the student employment server
***\n");

//student_employment_server_1_readStudentRecord(P,S):-
//                      student_employment_server_1_se(S,_,_).

//student_employment_server_1_saveStudentRecord(P,S,SFR).

                prologBuffer.append(componentName+"_readStudentRecord(P,S):- ");
                prologBuffer.append(componentName+"_se(S,_,_).\n");

                prologBuffer.append(componentName+"_saveStudentRecord(P,S,AR).\n\n");
        }
        else if (componentID.equals("RecordServer2"))
        {

                prologBuffer.append("%*** These facts represent student data in the record server
***\n");

                prologBuffer.append(componentName+"_sr(acrespi).\n");
                prologBuffer.append(componentName+"_sr(hblake).\n");
                prologBuffer.append(componentName+"_gsi(acrespi,alex,crespi,indianapolis).\n");
                prologBuffer.append(componentName+"_gsi(hblake,henry,blake,muncie).\n");
                prologBuffer.append(componentName+"_note(acrespi,fburns, ['this is a note']).\n");
                prologBuffer.append(componentName+"_note(acrespi,cwinchester,['this is another
note']).\n\n");

                prologBuffer.append("%*** Predicate for reading student record ***\n");

//record_server_2_readStudentRecord(P,S):-
//              financial_aid_server_1_readStudentRecord(P,S) ;
//              academic_record_server_1_readStudentRecord(P,S) ;
//              student_employment_server_1_readStudentRecord(P,S) ;
//              (record_server_1_gsi(S,_,_,_),record_server_1_Guard(P,[[sr,S],[section,gsi],[all,all]],read)).

                        prologBuffer.append(componentName+"_readStudentRecord(P,S):-\n");

                        //read in record server name from post processing collaborators list
                        String financialAidServerName = (String)
postProcessingCollaborators.get("financialaidserver");
                        String academicRecordServerName = (String)
postProcessingCollaborators.get("academicrecordserver");
                        String studentEmploymentServerName = (String)
postProcessingCollaborators.get("studentemploymentserver");

                        prologBuffer.append("    "+financialAidServerName     +"_readStudentRecord(P,S) ;\n");
                        prologBuffer.append("    "+academicRecordServerName   +"_readStudentRecord(P,S)
;\n");
                        prologBuffer.append("    "+studentEmploymentServerName+"_readStudentRecord(P,S)
;\n");
                        prologBuffer.append("    "+"("+componentName+"_gsi(S,_,_,_),");

            prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,gsi],[all,all]],read)).\n\n");


                        prologBuffer.append("%*** Predicate for writing student record ***\n");

//record_server_2_saveStudentRecord(P,S,SR):-
//          ((member(gsi,SR), record_server_1_Guard(P,[[sr,S],[section,gsi],[all,all]],save)) ; not
member(gsi,SR) ),
//          ((member(fa,SR) , financial_aid_server_1_saveStudentRecord(P,S,SR)); not member(fa,SR) ),
//          ((member(ar,SR) , academic_record_server_1_saveStudentRecord(P,S,SR)); not member(ar,SR)
),
//          ((member(se,SR) , student_employment_server_1_saveStudentRecord(P,S,SR)); not
member(se,SR) ).

                        prologBuffer.append(componentName+"_saveStudentRecord(P,S,SR):-\n");
                        prologBuffer.append("    "+"((member(gsi,SR),
"+componentName+"_Guard(P,[[sr,S],[section,gsi],[all,all]],save)) ; not member(gsi,SR) ),\n");
                        prologBuffer.append("    "+"((member(fa,SR) ,"+financialAidServerName
+"_saveStudentRecord(P,S,SR)); not member(fa,SR) ),\n");
                        prologBuffer.append("    "+"((member(ar,SR) ,"+academicRecordServerName
+"_saveStudentRecord(P,S,SR)); not member(ar,SR) ),\n");
                        prologBuffer.append("    "+"((member(se,SR)
,"+studentEmploymentServerName+"_saveStudentRecord(P,S,SR)); not member(se,SR) ).\n\n");


                        prologBuffer.append("%*** Predicate for record server guard ***\n");

//record_server_2_Guard(P,R,O):- record_server_1_DynamicAttributes(P,NEWP,R),
//                          accessControl(NEWP,R,O).

                        prologBuffer.append(componentName+"_Guard(P,R,O):-
"+componentName+"_DynamicAttributes(P,NEWP,R),\n");
                        prologBuffer.append("    "+"accessControl(NEWP,R,O).\n\n");
                        prologBuffer.append("%*** Predicate for record server dynamic attributes ***\n");
```

```
//record_server_2_DynamicAttributes(P,NEWP,R):-
//                      arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
//                      arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
//                      arg(3,P,GROUPPRED),
//                      (member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
//                      (member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
//                      NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).

                    prologBuffer.append(componentName+"_DynamicAttributes(P,NEWP,R):-\n");
                    prologBuffer.append("
"+"arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),\n");
                    prologBuffer.append("    "+"arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),\n");
                    prologBuffer.append("    "+"arg(3,P,GROUPPRED),\n");
                    prologBuffer.append("    "+"(member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),\n");
                    prologBuffer.append("    "+"(member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),\n");
                    prologBuffer.append("
"+"NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).\n");
          }
          else if (componentID.equals("AcademicRecordServer2"))
          {
                    prologBuffer.append("%*** These facts represent student data in the academic record
server ***\n");

          prologBuffer.append(componentName+"_ar(acrespi,dps([computer_science,business]),ts([[cs403,
a],[cs595,b]])).\n");

          prologBuffer.append(componentName+"_ar(hblake,dps([chemistry]),ts([[c101,a],[c132,a]])).\n\n"
);


                    prologBuffer.append("%*** These predicates represent the academic record server
***\n");

//academic_record_server_2_readStudentRecord(P,S):-
//                academic_record_server_1_ar(S,_,_),
//                academic_record_server_1_Guard(P,[[sr,S],[section,ar],[all,all]],read).

//academic_record_server_2_saveStudentRecord(P,S,AR):-
academic_record_server_1_Guard(P,[[sr,S],[section,ar],[all,all]],save).

                    prologBuffer.append(componentName+"_readStudentRecord(P,S):- ");
                    prologBuffer.append(componentName+"_ar(S,_,_),");
                    prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,ar],[all,all]],read).\n");

                    prologBuffer.append(componentName+"_saveStudentRecord(P,S,AR):- ");

          prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,ar],[all,all]],save).\n\n");

                    prologBuffer.append("%*** Predicate for academic record server guard ***\n");
```

```
//academic_record_server_2_Guard(P,R,O):- academic_record_server_1_DynamicAttributes(P,NEWP,R),
//                              accessControl(NEWP,R,O).

                    prologBuffer.append(componentName+"_Guard(P,R,O):-
"+componentName+"_DynamicAttributes(P,NEWP,R),\n");
                    prologBuffer.append("   "+"accessControl(NEWP,R,O).\n\n");

                    prologBuffer.append("%*** Predicate for academic record server dynamic attributes
***\n");


//academic_record_server_2_DynamicAttributes(P,NEWP,R):-
//                      arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
//                      arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
//                      arg(3,P,GROUPPRED),
//                      (member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
//                      (member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
//                      NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).

                    prologBuffer.append(componentName+"_DynamicAttributes(P,NEWP,R):-\n");
                    prologBuffer.append("
"+"arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),\n");
                    prologBuffer.append("   "+"arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),\n");
                    prologBuffer.append("   "+"arg(3,P,GROUPPRED),\n");
                    prologBuffer.append("   "+"(member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),\n");
                    prologBuffer.append("   "+"(member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),\n");
                    prologBuffer.append("
"+"NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).\n");
          }

          else if (componentID.equals("FinancialAidServer2"))
          {
                    prologBuffer.append("%*** These facts represent student data in the financial aid server
***\n");

          prologBuffer.append(componentName+"_fa(acrespi,[bankofamerica,sallemay],[pell],[volleyball],[
]).\n\n");

          prologBuffer.append(componentName+"_fa(hblake,[bankofamerica,sallemay],[pell],[volleyball],[
]).\n\n");


//financial_aid_server_2_readStudentRecord(P,S):-
//                      financial_aid_server_1_fa(S,_,_,_,_),
//                      financial_aid_server_1_Guard(P,[[sr,S],[section,fa],[all,all]],read).

//financial_aid_server_2_saveStudentRecord(P,S,SLR):-
//                      financial_aid_server_1_Guard(P,[[sr,S],[section,fa],[all,all]],save).
                    prologBuffer.append("%*** These predicates represent the financial aid server ***\n");
```

```
                prologBuffer.append(componentName+"_readStudentRecord(P,S):- ");
                prologBuffer.append(componentName+"_fa(S,_,_,_,_),");
                prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,fa],[all,all]],read).\n");

                prologBuffer.append(componentName+"_saveStudentRecord(P,S,AR):- ");

        prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,fa],[all,all]],save).\n\n");


                prologBuffer.append("%*** Predicate for academic financial aid guard ***\n");

//financial_aid_server_2_Guard(P,R,O):- financial_aid_server_1_DynamicAttributes(P,NEWP,R),
//                              accessControl(NEWP,R,O).

                prologBuffer.append(componentName+"_Guard(P,R,O):-
"+componentName+"_DynamicAttributes(P,NEWP,R),\n");
                prologBuffer.append("    "+"accessControl(NEWP,R,O).\n\n");

                prologBuffer.append("%*** Predicate for academic financial aid dynamic attributes
***\n");


//financial_aid_server_2_DynamicAttributes(P,NEWP,R):-
//                      arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
//                      arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
//                      arg(3,P,GROUPPRED),
//                      (member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
//                      (member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
//                      NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).

                prologBuffer.append(componentName+"_DynamicAttributes(P,NEWP,R):-\n");
                prologBuffer.append("
"+"arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),\n");
                prologBuffer.append("    "+"arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),\n");
                prologBuffer.append("    "+"arg(3,P,GROUPPRED),\n");
                prologBuffer.append("    "+"(member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),\n");
                prologBuffer.append("    "+"(member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),\n");
                prologBuffer.append("
"+"NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).\n");
        }

        else if (componentID.equals("StudentEmploymentServer2"))
        {
                prologBuffer.append("%*** These facts represent student data in the student
employment server ***\n");
        prologBuffer.append(componentName+"_se(acrespi,[[fall2003],[spring2004],[fall2004]],[]).\n\n")
;
        prologBuffer.append(componentName+"_se(hblake,[[fall2003],[spring2004],[fall2004]],[]).\n\n");
```

```
            prologBuffer.append("%*** These predicates represent the student employment server ***\n");

//student_employment_server_2_readStudentRecord(P,S):-
//                      student_employment_server_1_se(S,_,_),
//                      student_employment_server_1_Guard(P,[[sr,S],[section,se],[all,all]],read).

//student_employment_server_2_saveStudentRecord(P,S,SFR):-
//                      student_employment_server_1_Guard(P,[[sr,S],[section,se],[all,all]],save).

                        prologBuffer.append(componentName+"_readStudentRecord(P,S):- ");
                        prologBuffer.append(componentName+"_se(S,_,_),");
                        prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,se],[all,all]],read).\n");

                        prologBuffer.append(componentName+"_saveStudentRecord(P,S,AR):- ");

            prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,se],[all,all]],save).\n\n");


                        prologBuffer.append("%*** Predicate for student employment server guard ***\n");

//student_employment_server_2_Guard(P,R,O):-
student_employment_server_1_DynamicAttributes(P,NEWP,R),
//                              accessControl(NEWP,R,O).

                        prologBuffer.append(componentName+"_Guard(P,R,O):-
"+componentName+"_DynamicAttributes(P,NEWP,R),\n");
                        prologBuffer.append("    "+"accessControl(NEWP,R,O).\n\n");

                        prologBuffer.append("%*** Predicate for student employment server dynamic attributes
***\n");


//student_employment_server_2_DynamicAttributes(P,NEWP,R):-
//                        arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
//                        arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
//                        arg(3,P,GROUPPRED),
//                        (member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
//                        (member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
//                        NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).

                        prologBuffer.append(componentName+"_DynamicAttributes(P,NEWP,R):-\n");
                        prologBuffer.append("
"+"arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),\n");
                        prologBuffer.append("    "+"arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),\n");
                        prologBuffer.append("    "+"arg(3,P,GROUPPRED),\n");
                        prologBuffer.append("    "+"(member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),\n");
                        prologBuffer.append("    "+"(member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),\n");
                        prologBuffer.append("
"+"NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).\n");
```

```
            }


else if (componentID.equals("RecordServer3"))
            {

                        prologBuffer.append("%*** These facts represent student data in the record server
***\n");

                        prologBuffer.append(componentName+"_sr(acrespi).\n");
                        prologBuffer.append(componentName+"_sr(hblake).\n");
                        prologBuffer.append(componentName+"_gsi(acrespi,alex,crespi,indianapolis).\n");
                        prologBuffer.append(componentName+"_gsi(hblake,henry,blake,muncie).\n");
                        prologBuffer.append(componentName+"_note(acrespi,fburns, ['this is a note']).\n");
                        prologBuffer.append(componentName+"_note(acrespi,cwinchester,['this is another
note']).\n\n");


            prologBuffer.append(componentName+"_ar_cache(acrespi,dps([computer_science,business]),ts([[
cs403,a],[cs595,b]])).\n");

            prologBuffer.append(componentName+"_ar_cache(hblake,dps([chemistry]),ts([[c101,a],[c132,a]])
).\n");

            prologBuffer.append(componentName+"_fa_cache(acrespi,[bankofamerica,sallemay],[pell],[volle
yball],[]).\n");

            prologBuffer.append(componentName+"_fa_cache(hblake,[bankofamerica,sallemay],[pell],[volle
yball],[]).\n");

            prologBuffer.append(componentName+"_se_cache(acrespi,[[fall2003],[spring2004],[fall2004]],[])
.\n");

            prologBuffer.append(componentName+"_se_cache(hblake,[[fall2003],[spring2004],[fall2004]],[])
.\n\n");

                        prologBuffer.append("%*** Predicate for reading student record ***\n");

//record_server_3_readStudentRecord(P,S):-
//          (record_server_3_fa_cache(S,_,_,_,_);financial_aid_server_1_readStudentRecord(P,S) );
//          (record_server_3_ar_cache(S,_,_);academic_record_server_1_readStudentRecord(P,S) );
//          (record_server_3_se_cache(S,_,_);student_employment_server_1_readStudentRecord(P,S) ) ;
//          (record_server_1_gsi(S,_,_,_),record_server_1_Guard(P,[[sr,S],[section,gsi],[all,all]],read)).

                        prologBuffer.append(componentName+"_readStudentRecord(P,S):-\n");

                        //read in record server name from post processing collaborators list
                        String financialAidServerName = (String)
postProcessingCollaborators.get("financialaidserver");
                        String academicRecordServerName = (String)
postProcessingCollaborators.get("academicrecordserver");
                        String studentEmploymentServerName = (String)
postProcessingCollaborators.get("studentemploymentserver");
```

```
                    prologBuffer.append("   ("+componentName     +"_fa_cache(S,_,_,_,_);");
                    prologBuffer.append(financialAidServerName     +"_readStudentRecord(P,S) );\n");
                    prologBuffer.append("   ("+componentName     +"_ar_cache(S,_,_);");
                    prologBuffer.append(academicRecordServerName   +"_readStudentRecord(P,S) );\n");
                    prologBuffer.append("   ("+componentName     +"_se_cache(S,_,_);");
                    prologBuffer.append(studentEmploymentServerName+"_readStudentRecord(P,S) );\n");
                    prologBuffer.append("   "+"("+componentName+"_gsi(S,_,_,_),");

          prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,gsi],[all,all]],read)).\n\n");

                    prologBuffer.append("%*** Predicate for writing student record ***\n");

//record_server_3_saveStudentRecord(P,S,SR):-
//          ((member(gsi,SR), record_server_1_Guard(P,[[sr,S],[section,gsi],[all,all]],save)) ; not
member(gsi,SR) ),
//          ((member(fa,SR) , financial_aid_server_1_saveStudentRecord(P,S,SR)); not member(fa,SR) ),
//          ((member(ar,SR) , academic_record_server_1_saveStudentRecord(P,S,SR)); not member(ar,SR)
),
//          ((member(se,SR) , student_employment_server_1_saveStudentRecord(P,S,SR)); not
member(se,SR) ).

                    prologBuffer.append(componentName+"_saveStudentRecord(P,S,SR):-\n");
                    prologBuffer.append("   "+"((member(gsi,SR),
"+componentName+"_Guard(P,[[sr,S],[section,gsi],[all,all]],save)) ; not member(gsi,SR) ),\n");
                    prologBuffer.append("   "+"((member(fa,SR) ,"+financialAidServerName
+"_saveStudentRecord(P,S,SR)); not member(fa,SR) ),\n");
                    prologBuffer.append("   "+"((member(ar,SR) ,"+academicRecordServerName
+"_saveStudentRecord(P,S,SR)); not member(ar,SR) ),\n");
                    prologBuffer.append("   "+"((member(se,SR)
,"+studentEmploymentServerName+"_saveStudentRecord(P,S,SR)); not member(se,SR) ).\n\n");


                    prologBuffer.append("%*** Predicate for record server guard ***\n");

//record_server_3_Guard(P,R,O):- record_server_1_DynamicAttributes(P,NEWP,R),
//                    accessControl(NEWP,R,O).

                    prologBuffer.append(componentName+"_Guard(P,R,O):-
"+componentName+"_DynamicAttributes(P,NEWP,R),\n");
                    prologBuffer.append("   "+"accessControl(NEWP,R,O).\n\n");


                    prologBuffer.append("%*** Predicate for record server dynamic attributes ***\n");

//record_server_3_DynamicAttributes(P,NEWP,R):-
//                    arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
//                    arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
//                    arg(3,P,GROUPPRED),
//                    (member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
//                    (member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
//                    NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).
```

```
                        prologBuffer.append(componentName+"_DynamicAttributes(P,NEWP,R):-\n");
                        prologBuffer.append("
"+"arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),\n");
                        prologBuffer.append("   "+"arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),\n");
                        prologBuffer.append("   "+"arg(3,P,GROUPPRED),\n");
                        prologBuffer.append("   "+"(member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),\n");
                        prologBuffer.append("   "+"(member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),\n");
                        prologBuffer.append("
"+"NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).\n");
            }

else if (componentID.equals("RecordServer4"))
            {

                        prologBuffer.append("%*** These facts represent student data in the record server
***\n");

                        prologBuffer.append(componentName+"_sr(acrespi).\n");
                        prologBuffer.append(componentName+"_sr(hblake).\n");
                        prologBuffer.append(componentName+"_gsi(acrespi,alex,crespi,indianapolis).\n");
                        prologBuffer.append(componentName+"_gsi(hblake,henry,blake,muncie).\n");
                        prologBuffer.append(componentName+"_note(acrespi,fburns, ['this is a note']).\n");
                        prologBuffer.append(componentName+"_note(acrespi,cwinchester,['this is another
note']).\n\n");


            prologBuffer.append(componentName+"_ar_cache(acrespi,dps([computer_science,business]),ts([[
cs403,a],[cs595,b]])).\n");

            prologBuffer.append(componentName+"_ar_cache(hblake,dps([chemistry]),ts([[c101,a],[c132,a]])
).\n");

            prologBuffer.append(componentName+"_fa_cache(acrespi,[bankofamerica,sallemay],[pell],[volle
yball],[]).\n");

            prologBuffer.append(componentName+"_fa_cache(hblake,[bankofamerica,sallemay],[pell],[volle
yball],[]).\n");

            prologBuffer.append(componentName+"_se_cache(acrespi,[[fall2003],[spring2004],[fall2004]],[])
.\n");

            prologBuffer.append(componentName+"_se_cache(hblake,[[fall2003],[spring2004],[fall2004]],[])
.\n\n");

                        prologBuffer.append("%*** Predicate for reading student record ***\n");

//record_server_4_readStudentRecord(P,S):-
//((record_server_3_fa_cache(S,_,_,_,_),record_server_4_Guard(P,[[sr,S],[section,fa],[all,all]],read));financ
ial_aid_server_1_readStudentRecord(P,S) );
//((record_server_3_ar_cache(S,_,_),record_server_4_Guard(P,[[sr,S],[section,ar],[all,all]],read));academic
_record_server_1_readStudentRecord(P,S) );
```

```
//((record_server_3_se_cache(S,_,_),record_server_4_Guard(P,[[sr,S],[section,se],[all,all]],read));student_e
mployment_server_1_readStudentRecord(P,S) ) ;
//(record_server_1_gsi(S,_,_,_),record_server_4_Guard(P,[[sr,S],[section,gsi],[all,all]],read)).

                    prologBuffer.append(componentName+"_readStudentRecord(P,S):-\n");

                    //read in record server name from post processing collaborators list
                    String financialAidServerName = (String)
postProcessingCollaborators.get("financialaidserver");
                    String academicRecordServerName = (String)
postProcessingCollaborators.get("academicrecordserver");
                    String studentEmploymentServerName = (String)
postProcessingCollaborators.get("studentemploymentserver");

                    prologBuffer.append("   (("+componentName+"_fa_cache(S,_,_,_,_),");
                    prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,fa],[all,all]],read));");
                    prologBuffer.append(financialAidServerName+"_readStudentRecord(P,S) );\n");
                    prologBuffer.append("   (("+componentName+"_ar_cache(S,_,_),");
                    prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,ar],[all,all]],read));");
                    prologBuffer.append(academicRecordServerName+"_readStudentRecord(P,S) );\n");
                    prologBuffer.append("   (("+componentName+"_se_cache(S,_,_),");
                    prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,se],[all,all]],read));");
                    prologBuffer.append(studentEmploymentServerName+"_readStudentRecord(P,S) );\n");
                    prologBuffer.append("   "+"("+componentName+"_gsi(S,_,_,_),");

          prologBuffer.append(componentName+"_Guard(P,[[sr,S],[section,gsi],[all,all]],read)).\n\n");


                    prologBuffer.append("%*** Predicate for writing student record ***\n");

//record_server_4_saveStudentRecord(P,S,SR):-
//        ((member(gsi,SR), record_server_1_Guard(P,[[sr,S],[section,gsi],[all,all]],save)) ; not
member(gsi,SR) ),
//        ((member(fa,SR) , financial_aid_server_1_saveStudentRecord(P,S,SR)); not member(fa,SR) ),
//        ((member(ar,SR) , academic_record_server_1_saveStudentRecord(P,S,SR)); not member(ar,SR)
),
//        ((member(se,SR) , student_employment_server_1_saveStudentRecord(P,S,SR)); not
member(se,SR) ).

                    prologBuffer.append(componentName+"_saveStudentRecord(P,S,SR):-\n");
                    prologBuffer.append("   "+"((member(gsi,SR),
"+componentName+"_Guard(P,[[sr,S],[section,gsi],[all,all]],save)) ; not member(gsi,SR) ),\n");
                    prologBuffer.append("   "+"((member(fa,SR) ,"+financialAidServerName
+"_saveStudentRecord(P,S,SR)); not member(fa,SR) ),\n");
                    prologBuffer.append("   "+"((member(ar,SR) ,"+academicRecordServerName
+"_saveStudentRecord(P,S,SR)); not member(ar,SR) ),\n");
                    prologBuffer.append("   "+"((member(se,SR)
,"+studentEmploymentServerName+"_saveStudentRecord(P,S,SR)); not member(se,SR) ).\n\n");


                    prologBuffer.append("%*** Predicate for record server guard ***\n");

//record_server_4_Guard(P,R,O):- record_server_1_DynamicAttributes(P,NEWP,R),
//                        accessControl(NEWP,R,O).
```

```
			prologBuffer.append(componentName+"_Guard(P,R,O):-
"+componentName+"_DynamicAttributes(P,NEWP,R),\n");
			prologBuffer.append("    "+"accessControl(NEWP,R,O).\n\n");


			prologBuffer.append("%*** Predicate for record server dynamic attributes ***\n");

//record_server_4_DynamicAttributes(P,NEWP,R):-
//				arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),
//				arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),
//				arg(3,P,GROUPPRED),
//				(member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),
//				(member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),
//				NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).

			prologBuffer.append(componentName+"_DynamicAttributes(P,NEWP,R):-\n");
			prologBuffer.append("
"+"arg(1,P,AIDPRED),arg(1,AIDPRED,AID),nl,write(AID),\n");
			prologBuffer.append("    "+"arg(2,P,ROLEPRED),arg(1,ROLEPRED,ROLE),\n");
			prologBuffer.append("    "+"arg(3,P,GROUPPRED),\n");
			prologBuffer.append("    "+"(member([sr,AID],R)-
>append(ROLE,[studentowner],ROLE2);append(ROLE,[],ROLE2)),\n");
			prologBuffer.append("    "+"(member([noteowner,AID],R)-
>append(ROLE2,[noteowner],ROLE3);append(ROLE2,[],ROLE3)),\n");
			prologBuffer.append("
"+"NEWP=credentials(AIDPRED,role(ROLE3),GROUPPRED).\n");
		}

else {
			prologBuffer.append("% <<<NO MODEL FOR THIS COMPONENT.>>>\n\n");
		}
		return prologBuffer.toString();

}


// ADD IN A MAIN METHOD TO TEST OUT THIS CLASS
public static void main(String []args){

		PrologACModel model=new PrologACModel();
		try{
			PrintWriter pw=new PrintWriter(new FileWriter(new
File("studentRecordPrologACModel.pl")));
			// PRINT PROLOG FOR GENERAL SYSTEM PORTION
			String sm=model.getSystemModel();
			pw.println(sm);

			// PRINT PROLOG FOR TERMINAL
			//CREATE HASH TABLE OF PPC
			Hashtable ppc=new Hashtable();
			ppc.put("recordserver","record_server");
```

```
                    String ut=model.getComponentModel("user_terminal_1","user_terminal",ppc);
                    pw.println(ut);



                    // PRINT PROLOG FOR RECORD SERVER
                    //CREATE HASH TABLE OF PPC
                    ppc=new Hashtable();
                    ppc.put("financialaidserver","financial_server");
                    ppc.put("academicrecordserver","academic_server");
                    ppc.put("studentemploymentserver","employment_server");

                    String srs=model.getComponentModel("record_server_1", "record_server",ppc);
                    pw.println(srs);


                    // PRINT PROLOG FOR ACADEMIC RECORD SERVER
                    //CREATE HASH TABLE OF PPC
                    ppc=new Hashtable();

                    String
ars=model.getComponentModel("academic_record_server_1","academic_server",ppc);
                    pw.println(ars);

                    // PRINT PROLOG FOR FINANCIAL AID SERVER
                    //CREATE HASH TABLE OF PPC
                    ppc=new Hashtable();

                    String fas=model.getComponentModel("financial_aid_server_1","financial_server",ppc);
                    pw.println(fas);

                    // PRINT PROLOG FOR STUDENT EMPLOYMENT SERVER
                    //CREATE HASH TABLE OF PPC
                    ppc=new Hashtable();

                    String
ses=model.getComponentModel("student_employment_server_1","employment_server",ppc);
                    pw.println(ses);

                    pw.close();
            }catch(IOException e){System.out.println("ERROR:"+e);}

}
}
```

Simulator.java

This class coordinates the simulation of a headhunter searching for components and also generate the Prolog access control model for a selected system.

```
import java.security.*;
import java.io.*;
import java.lang.*;
import java.util.*;

/**
 * Insert the type's description here.
 * Creation date: (10/15/2001 11:50:10 AM)
 * @author: Nanditha Nayani
 */

public class Simulator{

public static QueryBean loadQueryBean(String ummFileName)
{
        QueryBean query;

        System.out.println("  Loading System Requirements");

        //Call the XMLParser by passing this URL. The XML Parser will parse the XML specification
        //and construct a Component from the specification which it returns.

        UniFrameQuerySpecificationParser xmlDomParser = new
UniFrameQuerySpecificationParser(ummFileName);
System.out.println("here");
        AbstractComponent component = xmlDomParser.getAbstractComponent();

        //Create query bean from component
        query = new QueryBean(component);


        System.out.println("  Done Loading Requirements.");

         //Once the Hashtable is filled return the hashtable
         return query;
 }


public static void main(String[] args) {

        long mcastTime = 50000;
        String headhunterLocation ="n/a";
        String dsmLocation="n/a";
        int mcastPort = 10000;
        String userName = "AC";
        String password = "simulation";
        String domain = "n/a";


        Headhunter h=new Headhunter(
                        mcastTime,
                        mcastPort,
                        userName,
                        password,
                        domain,
                        headhunterLocation,
                        dsmLocation);



        //SEARCH FOR MATCHING FINANCIAL AID SERVERS
        QueryBean q=Simulator.loadQueryBean("query2_financial_aid_server.xml");
        Hashtable financialServers=h.performSearch(q);


        //SEARCH FOR MATCHING ACADEMIC RECORD SERVERS
        q=Simulator.loadQueryBean("query2_academic_server.xml");
        Hashtable academicServers=h.performSearch(q);


        //SEARCH FOR MATCHING RECORD SERVERS
        q=Simulator.loadQueryBean("query2_record_server.xml");
        Hashtable recordServers=h.performSearch(q);


        //SEARCH FOR MATCHING USER TERMINALS
        q=Simulator.loadQueryBean("query2_user_terminal.xml");
        Hashtable userTerminals=h.performSearch(q);


        //SEARCH FOR MATCHING STUDENT EMPLOYMENT SERVERS
        q=Simulator.loadQueryBean("query2_student_employment_server.xml");
        Hashtable employmentServers=h.performSearch(q);


        //SELECT COMPONENTS
        ConcreteComponent userTerminal     =
(ConcreteComponent)userTerminals.get("UserTerminal1");
        ConcreteComponent recordServer     = (ConcreteComponent)recordServers.get("RecordServer3");
        ConcreteComponent academicServer   =
(ConcreteComponent)academicServers.get("AcademicRecordServer2");
        ConcreteComponent financialServer  =
(ConcreteComponent)financialServers.get("FinancialAidServer2");
        ConcreteComponent employmentServer =
(ConcreteComponent)employmentServers.get("StudentEmploymentServer2");


        //BUILD PROLOG MODEL FOR THE SYSTEM.
        PrologACModel model=new PrologACModel();
        try{
```

```
			PrintWriter pw=new PrintWriter(new FileWriter(new
File("studentRecordPrologACModel.pl")));

			String sm=model.getSystemModel();
			pw.println(sm);

			Hashtable ppc=new Hashtable();
			ppc.put("recordserver","record_server");
			String ut=model.getComponentModel(userTerminal.getID(),"user_terminal",ppc);
			pw.println(ut);

			ppc=new Hashtable();
			ppc.put("financialaidserver","financial_server");
			ppc.put("academicrecordserver","academic_server");
			ppc.put("studentemploymentserver","employment_server");
			String srs=model.getComponentModel(recordServer.getID(), "record_server",ppc);
			pw.println(srs);

			ppc=new Hashtable();
			String ars=model.getComponentModel(academicServer.getID(),"academic_server",ppc);
			pw.println(ars);

			ppc=new Hashtable();
			String fas=model.getComponentModel(financialServer.getID(),"financial_server",ppc);
			pw.println(fas);

			ppc=new Hashtable();
			String
ses=model.getComponentModel(employmentServer.getID(),"employment_server",ppc);
			pw.println(ses);

			pw.close();

			System.out.println("The PROLOG access control model is stored in:
studentRecordPrologACModel.pl");

		}catch(IOException e){System.out.println("ERROR:"+e);}

}

}
```

Appendix H: Component Test Results for Case Study

System 1 - No Resource Protected

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE
acrespi	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
acrespi	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
hblake	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
hblake	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
fburns	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
fburns	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
cwinchester	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
cwinchester	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE	RESULT
All users	SAVE	All students	YES/NO	YES/NO	YES/NO	YES/NO	SUCCEEDED

System 2 - All Resources Protected

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE
acrespi	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
acrespi	READ	hblake	BLOCKED	BLOCKED	BLOCKED	BLOCKED
hblake	READ	acrespi	BLOCKED	BLOCKED	BLOCKED	BLOCKED
hblake	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
fburns	READ	acrespi	ACCESSED	BLOCKED	ACCESSED	ACCESSED
fburns	READ	hblake	ACCESSED	BLOCKED	ACCESSED	ACCESSED
cwinchester	READ	acrespi	ACCESSED	ACCESSED	BLOCKED	BLOCKED
cwinchester	READ	hblake	ACCESSED	ACCESSED	BLOCKED	BLOCKED

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE	RESULT
acrespi	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
acrespi	SAVE	acrespi	YES	NO	NO	NO	SUCCEEDED
acrespi	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	hblake	YES	NO	NO	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	NO	YES	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	YES	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	YES	YES	SUCCEEDED
fburns	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
fburns	SAVE	hblake	NO	NO	NO	YES	SUCCEEDED
fburns	SAVE	hblake	NO	NO	YES	NO	SUCCEEDED
fburns	SAVE	hblake	NO	NO	YES	YES	SUCCEEDED
cwinchester	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
cwinchester	SAVE	acrespi	NO	YES	NO	NO	SUCCEEDED
cwinchester	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
cwinchester	SAVE	hblake	NO	YES	NO	NO	SUCCEEDED

System 3 - All Resources Protected

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE
acrespi	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
acrespi	READ	hblake	BLOCKED	BLOCKED	BLOCKED	BLOCKED
hblake	READ	acrespi	BLOCKED	BLOCKED	BLOCKED	BLOCKED
hblake	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
fburns	READ	acrespi	ACCESSED	BLOCKED	ACCESSED	ACCESSED
fburns	READ	hblake	ACCESSED	BLOCKED	ACCESSED	ACCESSED
cwinchester	READ	acrespi	ACCESSED	ACCESSED	BLOCKED	BLOCKED
cwinchester	READ	hblake	ACCESSED	ACCESSED	BLOCKED	BLOCKED

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE	RESULT
acrespi	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
acrespi	SAVE	acrespi	YES	NO	NO	NO	SUCCEEDED
acrespi	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	hblake	YES	NO	NO	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	NO	YES	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	YES	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	YES	YES	SUCCEEDED
fburns	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
fburns	SAVE	hblake	NO	NO	NO	YES	SUCCEEDED
fburns	SAVE	hblake	NO	NO	YES	NO	SUCCEEDED
fburns	SAVE	hblake	NO	NO	YES	YES	SUCCEEDED
cwinchester	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
cwinchester	SAVE	acrespi	NO	YES	NO	NO	SUCCEEDED
cwinchester	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
cwinchester	SAVE	hblake	NO	YES	NO	NO	SUCCEEDED

System 4 – Caches Subvert Read Access Control

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE
acrespi	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
acrespi	READ	hblake	BLOCKED	ACCESSED	ACCESSED	ACCESSED
hblake	READ	acrespi	BLOCKED	ACCESSED	ACCESSED	ACCESSED
hblake	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
fburns	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
fburns	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED
cwinchester	READ	acrespi	ACCESSED	ACCESSED	ACCESSED	ACCESSED
cwinchester	READ	hblake	ACCESSED	ACCESSED	ACCESSED	ACCESSED

PRINCIPAL	COMMAND	STUDENT	GSI	AR	FA	SE	RESULT
acrespi	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
acrespi	SAVE	acrespi	YES	NO	NO	NO	SUCCEEDED
acrespi	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
hblake	SAVE	hblake	YES	NO	NO	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	NO	YES	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	YES	NO	SUCCEEDED
fburns	SAVE	acrespi	NO	NO	YES	YES	SUCCEEDED
fburns	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
fburns	SAVE	hblake	NO	NO	NO	YES	SUCCEEDED
fburns	SAVE	hblake	NO	NO	YES	NO	SUCCEEDED
fburns	SAVE	hblake	NO	NO	YES	YES	SUCCEEDED
cwinchester	SAVE	acrespi	NO	NO	NO	NO	SUCCEEDED
cwinchester	SAVE	acrespi	NO	YES	NO	NO	SUCCEEDED
cwinchester	SAVE	hblake	NO	NO	NO	NO	SUCCEEDED
cwinchester	SAVE	hblake	NO	YES	NO	NO	SUCCEEDED